OPTIMASI SINKRONISASI DATA PADA ARSITEKTUR CQRS DENGAN KONTROL LQR

TESIS

Karya tulis sebagai salah satu syarat
Untuk memperoleh gelar magister dari
Institut Teknologi Bandung

Oleh:
DWI HANDOYO
NIM: 23524014
Program Studi Magister Informatika

INSTITUT TEKNOLOGI BANDUNG
MARET 2025

ABSTRAK

OPTIMASI SINKRONISASI DATA PADA ARSITEKTUR CQRS DENGAN KONTROL LQR

Oleh
Dwi Handoyo
NIM: 23524014
Program Studi Magister Informatika

Arsitektur *Command Query Responsibility Segregation* (CQRS) merupakan salah satu pendekatan yang umum digunakan dalam sistem terdistribusi untuk memisahkan basis data baca dan tulis. Meskipun memberikan keuntungan dalam hal skalabilitas, pendekatan ini juga membawa tantangan baru, khususnya dalam menjaga konsistensi data antar basis data. Tantangan ini semakin kompleks ketika proses sinkronisasi dilakukan secara asinkron melalui perantara *message broker*. Parameter-parameter seperti ukuran batch dan interval penarikan data perlu diatur secara tepat. Pengaturan batch yang besar akan mengoptimasi penggunaan daya namun menyebabkan waktu sinkronisasi menjadi lama. Untuk menyeimbangkan *trade-off* tersebut diperlukan sinkronisasi adaptif terhadap keadaan sistem.

Penelitian ini mengembangkan mekanisme pengendalian berbasis umpan balik menggunakan Linear Quadratic Regulator (LQR). Pendekatan ini memungkinkan penyesuaian parameter sinkronisasi secara otomatis dengan mempertimbangkan variable state, khususnya jumlah antrean pesan dalam message broker, serta tingkat pemakaian CPU dan memori pada server. Pengendali dirancang untuk meminimalkan fungsi biaya kuadratik yang mencakup dua variabel state tersebut dan juga variable kontrol yaitu *batch size* dan *poll interval*. Model sistem dinyatakan dalam bentuk *state-space* dan dianalisis sebagai sistem MIMO (*Multi-Input Multi-Output*). Pengujian dilakukan dengan membandingkan performa sistem pada skenario dengan pengaturan parameter statik dan pengendalian umpan balik.

Kata Kunci:
CQRS, sinkronisasi data, konsistensi, latency, Kafka, LQR, kontrol optimal, MIMO, state-space

OPTIMASI SINKRONISASI DATA PADA ARSITEKTUR CQRS DENGAN KONTROL LQR

TESIS

Oleh:
Dwi Handoyo
NIM: 23524014
Program Studi Magister Informatika

Institut Teknologi Bandung

Menyetujui Tim Pembimbing
Bandung, 2 Januari 2026
Ketua

Dr. Fazat Nur Azizah, S.T., M.Sc.
NIP. 197902102009122001

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR ISTILAH

Istilah	Nama	Halaman pertama muncul
CQRS	*Command Query Responsibility Segregation*, pola arsitektur yang memisahkan operasi baca dan tulis.	1
Open Loop	Eksperimen tanpa pengendalian untuk mengamati respon alami sistem terhadap input.	14
LQR	Linear Quadratic Regulator adalah sebuah metode kontrol optimal yang digunakan untuk sistem linier	2
PID	Pengendalian berbasis Proporsional, Integral dan Differensial.	8
PBS	Pendekatan probabilistik untuk mengukur kemungkinan baca data yang belum terubah.	4
Rise Time	Waktu yang dibutuhkan sistem untuk mencapai kondisi stabil atau target performa tertentu.	15

BAB I
PENDAHULUAN

I.1. Latar Belakang

Sistem Sistem terdistribusi modern menghadapi tantangan peningkatan volume data dan kebutuhan respon yang cepat seiring pertumbuhan pengguna dan layanan (Kleppmann, 2017). Untuk meningkatkan performa dan skalabilitas pada operasi baca dan tulis secara bersamaan, arsitektur Command Query Responsibility Segregation (CQRS) muncul sebagai salah satu pola yang banyak digunakan. CQRS memisahkan logika tulis (command) dan baca (query), memungkinkan tiap sisi dioptimalkan secara berbeda sesuai beban yang ditanggung (Fowler, 2011). Pola ini sangat relevan di lingkungan terdistribusi, di mana kebutuhan baca dan tulis sering kali berbeda signifikan dalam frekuensi dan kompleksitas (Munonye & Martinek, 2023).

Salah satu keunggulan nyata dari CQRS adalah efisiensi penggunaan sumber daya dalam operasi baca. Karena model baca dapat dirancang agar hanya mengambil field yang dibutuhkan saja, sistem tidak perlu memuat keseluruhan entitas data. Cara ini mengurangi overhead memori dan I/O, serta menghindari biaya tambahan yang muncul dari pemrosesan objek-objek besar ketika hanya sebagian kecil datanya yang dipakai (ZhangLong, 2017). Selain itu, dengan memisahkan model baca dan model tulis, beban tulis dapat diisolasi agar tidak mengganggu performa baca, misalnya mengurangi konflik lock database pada sistem tradisional di mana o ini memerlukan trasnformasi data sehingga memerlukan sumberdaya yang lebih besar dari pada arsitektur terdistribusi lainnya yang hanya berfokus pada sinkronisasi data.

Selain itu, pemisahan model ini membawa tantangan tersendiri, terutama dalam menjaga sinkronisasi antara basis data tulis dan baca. CQRS sering menggunakan mekanisme asinkron seperti event sourcing atau message broker untuk meneruskan event perubahan dari sisi tulis ke sisi baca (Morris et al., 2023). Pada mekanisme ini, muncul potensi eventual consistency, situasi di mana data pada model baca belum mencerminkan perubahan terbaru pada model tulis (Kleppmann, 2017). Meski inkonsistensi ini bersifat sementara, dalam

banyak aplikasi, terutama yang bersifat real-time atau kritikal, efeknya terhadap akurasi data tetap dapat signifikan (Vogels, 2009; Bailis et al., 2012).

Dalam praktik sinkronisasi berbasis event atau batch, pengaturan ukuran batch (batch size) menjadi aspek krusial. Di bawah beban tinggi, batch yang besar mampu mengurangi overhead per pesan seperti pengiriman jaringan, kompresi, atau penanganan log, sehingga throughput meningkat. Namun, batch besar juga menyebabkan latensi end-to-end lebih tinggi karena waktu tunggu pengisian batch dan frekuensi pengiriman yang lebih rendah. Selain itu, ketika batch besar mencapai sisi basis data baca atau tulis untuk sinkronisasi, bisa terjadi lonjakan penggunaan CPU, memori, dan I/O, yang pada gilirannya dapat menyebabkan penurunan performa throughput secara keseluruhan atau bahkan bottleneck. Sebaliknya, batch yang sangat kecil memang menurunkan latensi, tetapi justru menghasilkan overhead yang lebih besar karena frekuensi commit lebih sering, konsumsi CPU meningkat, dan beban jaringan bertambah (Wu et al., 2019; Rahman, 2010).

Hal ini sejalan dengan temuan Leonarczyk (2025) yang meneliti self-adaptive micro-batching pada sistem stream processing berbasis GPU. Penelitian tersebut menunjukkan bahwa ukuran batch memiliki pengaruh langsung terhadap konsumsi memori, efisiensi transfer data, dan latensi pemrosesan. Jika batch terlalu besar, maka sistem mengalami lonjakan penggunaan resource dan latensi yang signifikan, sedangkan batch terlalu kecil meningkatkan overhead penjadwalan. Studi pada *Micro-batch and data frequency for stream processing on multi-cores* oleh Adriano, dkk (2022) juga mengamati bahwa pada beban kerja dengan frekuensi data variatif, ukuran batch yang fleksibel membantu menjaga efisiensi CPU ketika beban puncak. Hasil ini memperkuat pandangan bahwa pengaturan ukuran batch yang adaptif merupakan kunci dalam menjaga keseimbangan antara performa dan efisiensi sumber daya

Beberapa penelitian terdahulu telah mencoba menangani trade-off ini melalui pendekatan dynamic batching. Wu, dkk (2020) mengusulkan reactive batching strategy pada Apache Kafka, yang menyesuaikan ukuran batch berdasarkan metrik sistem seperti latensi, throughput. Pendekatan ini efektif mengurangi lonjakan latensi, namun berfokus pada optimisasi ukuran batch dengan pertimbangan latency dan throughput. Pada penerapan

event sourching untuk CQRS, resource utilization juga perlu dipertimbangkan. Penelitian lain dilakukan oleh Das dkk. (2018) dengan memperkenalkan Structured Streaming pada Apache Spark, yang menggunakan micro-batching adaptif untuk menyeimbangkan kebutuhan latensi rendah dan efisiensi sumber daya. Sementara itu, Wu, dkk (2019) pada penelitian lain mengembangkan model teori antrian untuk memprediksi dampak konfigurasi Kafka, termasuk batch size, terhadap throughput dan latensi, meski hasilnya sebatas prediksi tanpa mekanisme kendali adaptif.

Berdasarkan penelitian-penelitian tersebut, dapat disimpulkan bahwa meskipun dynamic batching terbukti meningkatkan performa sistem, sebagian besar pendekatannya masih berbasis heuristik, prediksi sederhana, atau kontrol klasik. Penelitian ini akan menggunakan Linear Quadratic Regulator (LQR), sebuah metode kontrol optimal yang memungkinkan perumusan masalah sinkronisasi dalam kerangka state-space. Dengan LQR, trade-off antara konsistensi data dan efisiensi sumber daya dapat dimodelkan secara matematis dalam bentuk fungsi biaya kuadratik. Pendekatan ini diharapkan mampu memberikan solusi yang lebih sistematis dan optimal untuk menjaga konsistensi data sekaligus mengoptimalkan penggunaan sumber daya dalam arsitektur CQRS.

I.2. Masalah Penelitian

Permasalahan pada penelitian sebelumnya membuka kebutuhan akan pendekatan baru yang lebih adaptif. Sistem perlu memiliki kemampuan untuk menyesuaikan mekanisme sinkronisasi secara otomatis berdasarkan situasi aktual seperti seberapa penuh memori, seberapa banyak data, atau bagaimana kondisi lalu lintas data saat itu. Pendekatan seperti ini bisa dibangun menggunakan teori kontrol yaitu *Linear Quadratic Regulator* (LQR), yang memang dirancang untuk sistem yang berubah-ubah dan berbasis umpan balik.

Linear Quadratic Regulator (LQR) adalah salah satu metode kontrol dengan memanfaatkan ruang keadaan (*state space).* Ruang keadaan merepresentasikan hubungan variabel masukan dan variabel keadaan dalam bentuk matriks. Sistem kontrol LQR akan melakukan operasi matriks pada setiap variabel untuk menentukan nilai-nilai variabel input berdasarkan seberapa besar simpangan variabel state dengan targetOleh karena itu, diperlukan analisis terhadap hubungan antar variabel.

Permasalahan yang diangkat pada penelitian ini berkaitan dengan bagaimana merancang sistem pengendalian umpan balik yang berbasis *state-space*. Tujuannya adalah agar sistem mampu menyesuaikan parameter sinkronisasi secara otomatis berdasarkan kondisi sumber daya yang sedang berlangsung. Tantangan dalam hal ini terletak pada bagaimana membentuk model matematis dari sistem yang kompleks dan dinamis seperti CQRS, serta merancang pengendali yang mampu merespons perubahan kondisi dengan cepat.

Terakhir, penelitian ini juga mempertanyakan sejauh mana pendekatan pengendalian adaptif tersebut mampu meningkatkan performa dan menjaga konsistensi sistem dibandingkan dengan pendekatan statik yang lazim digunakan saat ini. Evaluasi ini penting untuk menilai kelayakan penerapan kontrol dinamis dalam skenario dunia nyata. Selain itu, hasil evaluasi juga akan menjadi dasar argumentasi apakah kontrol adaptif berbasis LQR benar-benar memberikan keunggulan yang signifikan terhadap sistem sinkronisasi berbasis event.

Berdasarkan latar belakang yang telah diuraikan diatas dapat disimpulkan rumusan masalah sebagai berikut:

1. Bagaimana perancangan pengendalian umpan balik berbasis *state space* yang mampu mengatur parameter sinkronisasi berdasarkan kondisi sumber daya secara real-time?
2. Sejauh mana efektivitas pendekatan tersebut dalam menjaga performa dan konsistensi sistem dibandingkan pendekatan statik?

I.3. Tujuan

Penelitian ini bertujuan untuk:

1. Membuat system yang mencapai keseimbangan antara konsistensi yang baik dengn waktu dan pemakaian sumber daya yang optimal
2. Merancang sistem pengendalian umpan balik yang adaptif berbasis kontrol umpan balik LQR yang mampu mengelola parameter sinkronisasi secara dinamis.

I.4. Batasan Masalah

Penelitian ini berfokus pada aspek fundamental dari mekanisme sinkronisasi data pada arsitektur berbasis event sourcing. Untuk menjaga fokus penelitian dan memastikan analisis

yang mendalam serta terukur, ruang lingkup penelitian dibatasi secara eksplisit sebagai pertimbangan metodologis, bukan karena keterbatasan implementasi.

Dalam penelitian ini digunakan satu server database tulis dan satu server database baca, serta satu consumer dan satu topic pada message broker. Pembatasan ini dilakukan untuk menghindari kompleksitas tambahan seperti heterogenitas replika, dinamika diskrit, dan perilaku switching yang muncul pada sistem dengan multiple read replica dan mekanisme quorum, sehingga dinamika sinkronisasi dapat direpresentasikan secara konsisten dalam kerangka pemodelan linear yang sesuai dengan pendekatan kontrol LQR.

Adapun batasan-batasan yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Terdapat satu server database baca dan satu server database tulis.
2. Terdapat satu *consumer* dan satu *topic* pada *message broker.*

I.5. Sistematika Penulisan

Pembahasan proposal penelitian ini akan terbagi dalam tiga bab, yaitu sebagai berikut:

1. Bab I Pendahuluan

   Bab ini berisi latar belakang, rumusan masalah, tujuan, batasan, hipotesis, metodologi, dan sistematika penulisan.

2. Bab 2 Tinjauan Pustaka

   Bab ini membahas teori dasar terkait arsitektur pemisahan baca-tulis, kontrol umpan balik, dan metrik evaluasi sistem.

3. Bab 3 Metode Penelitian

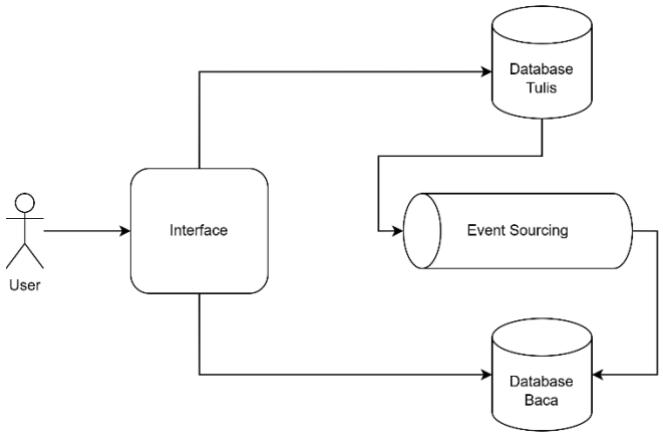
   Bab ini memaparkan pendekatan eksperimen, skenario pengujian, dan variabel yang digunakan.

BAB II
TINJAUAN PUSTAKA

II.1. Arsitektur Command Query Responsibility Segregation (CQRS)

Command Query Responsibility Segregation (CQRS) adalah pola desain arsitektur yang memisahkan tanggung jawab antara operasi yang mengubah data (*command*) dan operasi yang membaca data (*query*). Arsitektur ini memungkinkan pemisahan model data untuk *query* tulis dan baca, sehingga masing-masing *query* dapat dioptimalkan sesuai kebutuhan (Fowler, 2011).

Keunggulan utama penerapan CQRS diantaranya peningkatan modularitas, skalabilitas dan kinerja sistem, karena operasi baca dan tulis dapat dipisahkan (Fowler, 2011). Selain itu, pemisahan ini memungkinkan pengembangan dan pemeliharaan yang lebih mudah, karena tim dapat fokus pada aspek spesifik tanpa mempengaruhi keseluruhan sistem. Namun, tantangan yang muncul antara lain peningkatan kompleksitas arsitektur dan kebutuhan untuk mengelola konsistensi data antara model baca dan tulis. Kedua model data tersebut harus tersinkronisasi melalu event handler yang mengubah database baca (Munonye, 2020)



Gambar II.1 Arsitekur CQRS

Pada arsitektur master-slave, satu node utama (master) bertanggung jawab atas operasi tulis, sementara node lainnya (slave) mereplikasi data dan menangani operasi baca (Kleppman, 2017). Pendekatan ini meningkatkan ketersediaan dan skalabilitas baca, namun tidak memisahkan model data untuk operasi baca dan tulis, sehingga optimasi spesifik sulit dilakukan. Sebaliknya, CQRS memisahkan model data untuk operasi baca dan tulis,

memungkinkan optimasi yang lebih spesifik dan fleksibilitas dalam pengembangan fitur baru (Fowler, 2011). Namun karena itu, CQRS membutuhkan sumber daya yang lebih besar daripada arsitektur master-slave. Selain itu, manajemen konsistensi data yang lebih kompleks dibandingkan dengan arsitektur master-slave.

II.2. Sinkronisasi Data secara Asinkron

Pada arsitektur CQRS, sinkronisasi data menjadi salah satu tantangan yang paling sering dihadapi. Karena data yang ditulis ke database utama tidak langsung terkirim di database baca, sering kali terjadi perbedaan data dalam kurun waktu tertentu (Kleppmann, 2017). Untuk menjaga fleksibilitas dan performa sistem, proses sinkronisasi ini biasanya dilakukan secara asinkron. Artinya, data yang ditulis tidak langsung dikirim ke database baca dalam satu transaksi, melainkan melalui jalur terpisah, seperti *message queue*.

Keunggulan dari komunikasi asinkron ini adalah kemampuannya mengurangi waktu respons. Komponen pembaca tidak perlu menunggu proses penulisan selesai, sehingga sistem bisa tetap responsif. Selain itu, bila sistem penerima sedang tidak tersedia, data dapat terjaga karena akan disimpan di antrian hingga bisa diproses. Namun, pendekatan ini menyebabkan data tidak langsung tersinkron. Artinya, apa yang dibaca pengguna belum tentu data terbaru yang sebenarnya sudah ada di sistem penulis.

Masalah lainnya adalah ketidakpastian durasi yang dibutuhkan untuk sinkronisasi. Dalam praktiknya, hal ini bisa berdampak pada akurasi sistem. Misalnya, jika sebuah permintaan atau transaksi diproses sebelum data terbaru masuk ke database baca, hasilnya bisa jadi tidak sesuai dengan kondisi sebenarnya (Kleppmann, 2017).

Beberapa solusi telah dicoba untuk mengatasi isu ini, seperti menggunakan penanda waktu (timestamp), versioning, atau melakukan sinkronisasi secara berkala berdasarkan interval tertentu. Namun, pendekatan-pendekatan tersebut umumnya bersifat statis kurang fleksibel terhadap perubahan kondisi sistem, seperti ketika beban permintaan meningkat drastis atau sumber daya pembaca mulai terbatas.

Dari permasalahan tersebut, muncul kebutuhan akan pendekatan sinkronisasi yang lebih adaptif. Salah satu alternatifnya adalah dengan menerapkan prinsip kontrol umpan balik.

Dalam pendekatan ini, parameter sinkronisasi seperti seberapa sering data diambil atau seberapa besar data dikirim dapat disesuaikan secara otomatis berdasarkan kondisi nyata yang sedang terjadi. Dengan cara ini, sistem dapat mempertahankan keseimbangan antara konsistensi dan performa, tanpa perlu intervensi manual.

II.3. Pengendalian Umpan Balik dalam Sistem Perangkat Lunak Berbasis LQR

Pengendalian umpan balik merupakan pendekatan yang digunakan untuk mengatur perilaku sistem agar tetap stabil dan optimal dengan menyesuaikan parameter-parameter kontrol berdasarkan pengamatan kondisi sistem dan deviasi dengan nilai yang diinginkan secara real-time (Ogata, 2010). Dalam konteks sistem sinkronisasi basis data, pengendalian umpan balik memungkinkan sistem untuk mengatur aspek-aspek seperti konsumsi sumber daya atau laju eksekusi proses mengikuti perubahan kondisi lingkungan seperti beban kerja maupun *latency* dari infrastruktur.

Salah satu alasan pendekatan ini relevan dalam sistem perangkat lunak modern adalah karena sistem perangkat lunak modern semakin bersifat kompleks, sehingga konfigurasi secara static tidak dapat selalu menjadi yang optimal. Untuk mengatasi hal ini, digunakan sistem kontrol berbasis *feedback loop* yang dapat bereaksi terhadap deviasi dari performa yang diinginkan. Pengendalian seperti ini telah diterapkan pada pengaturan throughput layanan cloud, alokasi dinamis memori, dan pengendalian autoscaling sistem.

II.3.1. Teori Kontrol pada Sistem Perangkat Lunak

Teori kontrol merupakan disiplin ilmu yang membahas bagaimana suatu sistem dapat mengatur dirinya sendiri berdasarkan sinyal umpan balik (*feedback*). Secara prinsip, teori kontrol bekerja dengan cara mengukur deviasi antara kondisi aktual sistem dengan kondisi yang diharapkan (*setpoint*), lalu menghasilkan tindakan perbaikan untuk meminimalkan perbedaan tersebut (Ogata, 2010).

Penerapan teori kontrol dalam domain komputasi modern bukanlah hal baru. Studi oleh Armağan dan Sümer (2014) menunjukkan bagaimana sistem kontrol umpan balik dapat secara efektif mengatur penggunaan CPU dan memori pada server basis data virtual. Penelitian tersebut menggunakan model sistem berbasis *state-space* dan menerapkan pengendali SISO serta MIMO untuk menjaga kualitas layanan (QoS) di bawah beban kerja

yang berubah-ubah secara signifikan. Lebih lanjut, pendekatan adaptif seperti PID juga terbukti efektif dalam pengelolaan QoS di lingkungan komputasi awan. Dalam penelitian oleh Liu et al. (2018), pengendali PID adaptif diterapkan untuk menyesuaikan alokasi sumber daya berdasarkan fluktuasi beban secara real-time. Hasilnya menunjukkan peningkatan stabilitas dan efisiensi dibandingkan pendekatan statik

Dalam sistem perangkat lunak, sinyal umpan balik dapat berupa metrik runtime seperti penggunaan CPU, backlog antrian data, *response time*. Dengan menggunakan pengendali yang sesuai, sistem dapat mengubah parameter konfigurasinya, seperti ukuran batch, atau frekuensi polling, agar kinerja sistem tetap stabil meskipun kondisi eksternal berubah. Pendekatan seperti ini sangat relevan dalam arsitektur modern yang bersifat elastis dan skalabel, serta ketika sistem perlu menyeimbangkan antara dua tujuan yang saling bertentangan misalnya performa dan konsistensi data.

Model formal dari sistem yang dikendalikan biasanya dinyatakan dalam bentuk *state-space representation*, di mana sistem digambarkan sebagai hubungan linier antara status saat ini, aksi kontrol, dan status selanjutnya:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ y_t &= Cx_t + Du_t \end{aligned} \tag{II.1}$$

Dalam konteks sistem sinkronisasi data, status sistem *x* dapat berupa utilitas cpu, jumlah antrian pesan, atau utilisasi memori, sedangkan *u* adalah tindakan yang bisa diambil sistem seperti memperbesar batch atau memperpanjang interval polling. Matriks *A* dan *B* menggambarkan dinamika sistem, yaitu bagaimana sistem berubah sebagai akibat dari tindakan yang dilakukan. Sedangkan variabel pada vector *y* dapat termati secara langsung dan biasanya merupakan *subset* dari *x*.

Dengan pendekatan ini, kita tidak hanya bereaksi terhadap kondisi saat ini, tetapi juga mengoptimalkan tindakan agar sistem tetap berada dalam kondisi yang stabil dan efisien. Di sinilah pendekatan kontrol optimal seperti LQR menjadi sangat relevan.

II.3.1. Linear Quadratic Regulator

Linear Quadratic Regulator (LQR) adalah salah satu teknik pengendalian optimal yang terkenal dan banyak digunakan dalam dunia teknik kendali. Keunggulan utama dari LQR

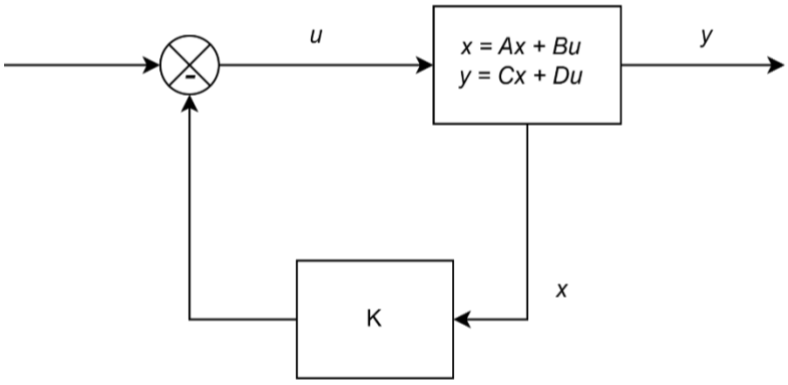
adalah kemampuannya untuk secara matematis menyeimbangkan antara dua tujuan yang sering bertentangan yaitu meminimalkan kesalahan sistem dan meminimalkan usaha kontrol.

Secara formal, LQR bertujuan untuk meminimalkan fungsi biaya kuadratik berikut ini:

$$J = \sum_{t=0}^{\infty} x_t^T \mathrm{Q} x_t + u_t^T R u_t \tag{II.2}$$

Di mana:

- J: fungsi biaya
- x_t: status sistem saat ini (misalnya, antrian pesan atau tingkat utilisasi CPU),
- u_t: sinyal kontrol (misalnya, pengaturan batch size, poll interval),
- Q: bobot penalti terhadap kesalahan status sistem, berupa matriks definit positif,
- R: bobot penalti terhadap usaha pengendalian, berupa matriks definit positif,.



Gambar II.2 Kontrol LQR

Solusi optimal dari fungsi biaya ini adalah hukum kontrol linier:

$$u_t = -K x_t \tag{II.3}$$

dengan *K* adalah matriks penguat (gain) yang diperoleh dari penyelesaian persamaan Riccati. Matriks ini menunjukkan seberapa besar tindakan korektif yang harus diambil untuk setiap kesalahan yang terdeteksi dalam status sistem.

Pendekatan LQR cocok digunakan untuk sistem yang kompleks, linear dan bersifat multi input dan multi output (MIMO). Dalam sistem sinkronisasi yang kompleks, kita mungkin perlu mengontrol beberapa parameter sekaligus seperti batch size dan interval polling. Keduanya menentukan multi variable kontrol dan tidak dapat diatur secara independen. Pada kasus seperti ini, LQR memberikan kerangka kerja matematis yang sistematis untuk merancang pengendali yang mempertimbangkan semua variabel sekaligus.

Namun, untuk menerapkan LQR, diperlukan pemodelan sistem terlebih dahulu. Hal ini dilakukan dengan cara melakukan eksperimen untuk merekam perubahan status sistem sebagai respons terhadap perubahan parameter kontrol, lalu menggunakan teknik estimasi seperti regresi linier atau metode identifikasi sistem (Ljung, 1999). Hasilnya berupa estimasi terhadap matriks A dan B, yang diperlukan untuk merancang kontroler LQR.

Walaupun implementasinya lebih kompleks dibandingkan pendekatan seperti PID, LQR menawarkan keunggulan dalam hal kestabilan, keoptimalan, dan fleksibilitas. Penentuan parameter kontrol pada LQR juga lebih sesuai untuk diterapkan pada kondisi berbasis *trade-off* karena didasarkan pada matriks biaya yang bisa diatur untuk menentukan variabel mana yang akan diunggulkan dibanding yang lain. Oleh karena itu, dalam sistem yang memerlukan kontrol presisi tinggi terhadap dinamika kompleks seperti dalam pengelolaan sinkronisasi data yang adaptif dan berbasis resource utilization LQR merupakan pilihan yang sangat menjanjikan.

II.4. Probabilistically Bounded Staleness (PBS) sebagai Metode Pengukuran Konsistensi

Konsistensi dalam sistem terdistribusi menjadi salah satu aspek fundamental dalam desain arsitektur berbasis event-driven dan distributed databases. Menurut Brewer (2000), CAP Theorem menyatakan bahwa dalam sistem terdistribusi, hanya dua dari tiga aspek berikut yang dapat dijamin secara bersamaan: consistency (C), availability (A), dan partition tolerance (P) (Brewer, 2000). Sistem yang mengutamakan konsistensi biasanya menggunakan replikasi secara sinkron untuk memastikan bahwa semua atau sebagian besar

node memiliki data yang sama pada waktu yang bersamaan, sementara sistem yang mengutamakan ketersediaan memungkinkan pembacaan dari node yang berbeda meskipun data belum direplikasi sepenuhnya.

Dalam konteks eventual consistency, data di seluruh node dalam sistem tidak harus konsisten secara real-time, tetapi akan mencapai keadaan yang konsisten setelah periode tertentu (Vogels, 2009).

II.4.1. Probabilistically Bounded Staleness (PBS) sebagai Metode Pengukuran Konsistensi

Probabilistically Bounded Staleness (PBS) adalah pendekatan kuantitatif untuk mengukur probabilitas suatu sistem mengalami stale reads. PBS mendefinisikan probabilitas P_stale, yang menunjukkan kemungkinan suatu permintaan baca mendapatkan data yang lebih lama dibandingkan dengan nilai terbaru yang telah diperbarui dalam sistem (Bailis, 2011).

Pendekatan PBS sering digunakan pada arsitektur yang menerapkan eventual consistency dan juga sistem yang berbasis event-driven menggunakan *message queue* untuk sinkroniasasi data. Dengan menganalisis distribusi latensi pembaruan data dari write database ke read database, metode ini memungkinkan pengukuran efektivitas konfigurasi sinkronisasi dalam sistem terdistribusi (Bailis, 2011).

Untuk mengukur P_stale secara praktis, dilakukan pengumpulan data terhadap latensi antar event. Latensi yang dimaksud adalah selisih waktu antara saat data berhasil ditulis di *write database* dan saat yang sama data tersebut tersedia untuk dibaca di *read database*. Dengan mengumpulkan data latensi dari banyak operasi tulis dan baca, kita bisa membangun distribusi empiris dari delay sinkronisasi. Selanjutnya, dengan memanfaatkan distribusi ini, dihitunglah probabilitas bahwa suatu permintaan baca terjadi dalam interval waktu sebelum pembaruan data tiba di *read database*. Probabilitas inilah yang digunakan sebagai estimasi nilai P_stale.

Nilai P_stale yang tinggi menunjukkan bahwa sistem sering kali memberikan data yang tidak mutakhir kepada pengguna, sedangkan nilai yang rendah menandakan bahwa mekanisme sinkronisasi berjalan dengan baik dan respons baca cenderung akurat. Dengan

demikian, PBS menjadi alat penting untuk mengevaluasi efektivitas konfigurasi sinkronisasi dalam sistem terdistribusi dan memberikan dasar kuantitatif untuk membandingkan berbagai strategi pengendalian atau parameter sistem yang diterapkan.

BAB III
ANALISIS DAN SOLUSI

Fenomena ketidaksinkronan pada arsitektur CQRS muncul sebagai konsekuensi dari pemisahan model tulis dan baca yang bergantung pada aliran event. Pada kondisi beban yang fluktuatif, laju kedatangan event tidak selalu sejalan dengan kapasitas pemrosesan konsumer, sehingga backlog dapat meningkat dan menghasilkan keterlambatan pembaruan pada database baca. Besarnya keterlambatan ini bergantung pada dinamika system. Diantaranya adalah laju produksi event, kapasitas pemrosesan, serta parameter seperti ukuran batch dan interval polling yang digunakan dalam pipeline sinkronisasi.

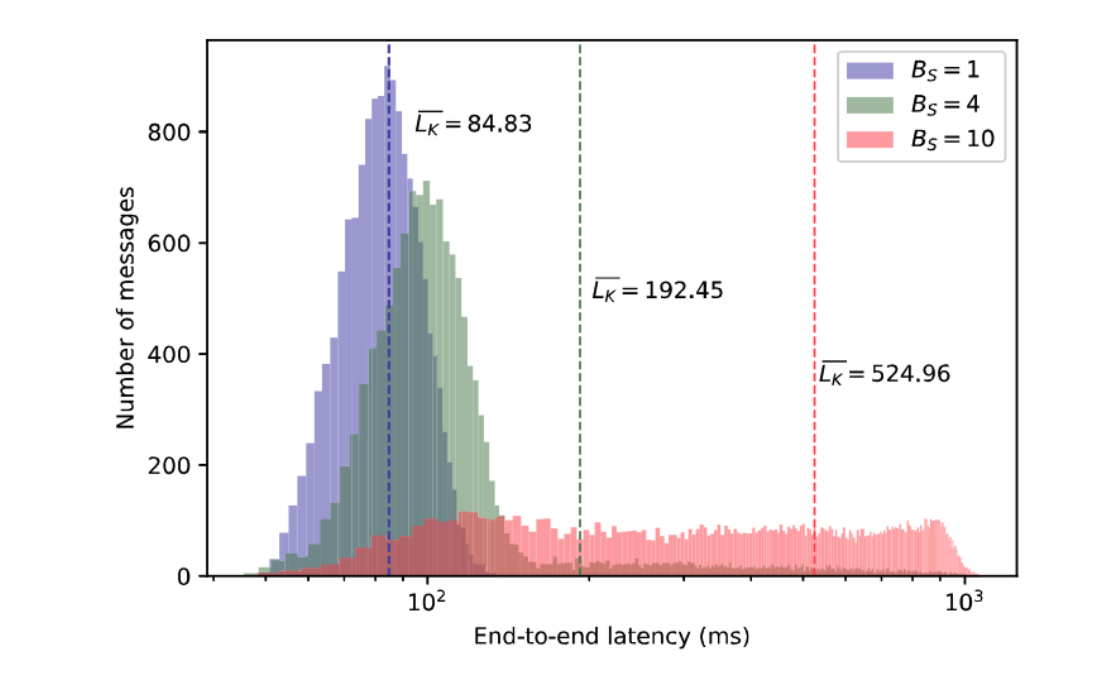
Variasi kondisi ini membutuhkan sistem yang bersifat dinamis. Selain itu, parameter kendali yang bersifat statis tidak selalu mampu mempertahankan performa yang stabil. Pada saat yang sama, peningkatan ukuran batch memang dapat menurunkan frekuensi pemrosesan dan mengurangi overhead, tetapi juga berpotensi menambah penundaan pembaruan data. Sebaliknya, ukuran batch yang terlalu kecil menghasilkan pemrosesan yang lebih responsif, namun cenderung meningkatkan konsumsi sumber daya dan menurunkan throughput.

Untuk itu, diperlukan system yang mampu menyeimbangkan kebutuhan akan data yang responsif namun juga efisien dalam penggunaan sumber daya. Dalam perancangan system tersebut, diperlukan pemahaman yang baik mengenai hubungan antara variabel-variabel yang membentuk dinamika sistem. Analisis ini mencakup identifikasi variabel state yang mewakili kondisi sistem pada suatu waktu, variabel kontrol yang dapat diatur, serta bagaimana perubahan pada variabel tersebut memengaruhi performa sinkronisasi, baik dalam hal backlog, latensi, maupun utilisasi sumber daya. Pemahaman ini menjadi dasar perumusan model pengendalian yang memungkinkan penyesuaian parameter secara adaptif agar sistem tetap berada pada kondisi yang stabil dan efisien.

III.1. Analisis Masalah

Alur sinkronisasi pada arsitektur CQRS bergantung pada proses konsumsi event dari log, pengelompokan event ke dalam batch, dan materialisasi ulang menjadi read model. Pada kondisi beban rendah, alur ini cenderung stabil karena kapasitas pemrosesan sejalan dengan laju kedatangan event. Namun ketika terjadi peningkatan throughput penulisan atau penurunan kapasitas konsumsi, backlog mulai meningkat dan menimbulkan keterlambatan propagasi data ke read model. Keterlambatan ini dapat secara langsung dirasakan pengguna dalam hal ketidakakuratan data karena data yang tidak aktual.

Masalah menjadi lebih kompleks karena pengaturan parameter utama seperti ukuran batch dan interval polling tidak selalu menaikan performa. Ukuran batch yang lebih besar dapat meningkatkan efisiensi pemrosesan karena mengurangi frekuensi commit dan overhead jaringan dan komputasi, tetapi juga menunda waktu mulai pemrosesan sehingga menambah latensi sinkronisasi. Sebaliknya, ukuran batch yang lebih kecil mempercepat pembaruan data tetapi meningkatkan konsumsi sumber daya CPU dan I/O, yang pada tingkat tertentu justru menurunkan throughput keseluruhan. Ketidakseimbangan ini menyebabkan sistem rentan terhadap ketidakpastian performa ketika beban berubah-ubah, terutama pada skenario bersifat bursty.



Gambar III.1 Pengaruh ukuran batch terhadap rata rata *latency* (Wu, 2020)

Untuk memahami perilaku sistem tersebut, beberapa variabel dapat dianggap sebagai representasi kondisi atau state sistem. Backlog event menjadi indikator utama yang menggambarkan selisih antara laju produksi dan laju konsumsi pada saat itu. Latensi pemrosesan event menunjukkan waktu yang dibutuhkan untuk memproses sebuah perubahan dan sangat dipengaruhi oleh besarnya batch, kapasitas komputasi, serta tingkat kontensi sumber daya. Selain itu, utilisasi CPU atau memori pada komponen konsumer juga merefleksikan kapasitas sistem dalam merespons beban tambahan. Variabel-variabel ini tidak hanya merepresentasikan kondisi sesaat, tetapi juga membentuk dinamika temporal yang menentukan apakah sistem akan stabil atau mengalami akumulasi backlog secara terus-menerus.

Di sisi lain, variabel yang dapat dikendalikan atau **c**ontrol variables meliputi ukuran batch dan interval polling pada konsumer. Kedua parameter ini secara langsung mengatur ritme konsumsi data dan besarnya unit kerja yang diproses dalam setiap siklus. Karena keduanya memengaruhi backlog, latensi, dan utilisasi sumber daya, kombinasi nilai yang tidak tepat

dapat menyebabkan kondisi sistem berada di luar rentang operasional yang diinginkan. Masalah inilah yang ingin diatasi melalui pendekatan pengendalian: menentukan nilai pengaturan yang dapat menyeimbangkan pergerakan variabel state agar tetap berada pada kondisi stabil meskipun terjadi variasi beban.

Namun hubungan antara variabel state dan kontrol bersifat dinamis dan tidak selalu mudah dipetakan secara langsung. Backlog, misalnya, bergantung pada interaksi antara laju produksi event dan kapasitas pemrosesan aktual pada konsumer, yang keduanya dapat bervariasi seiring waktu. Latensi pemrosesan batch dapat berubah akibat peningkatan jumlah thread aktif, perubahan beban komputasi, atau efek kontensi I/O. Ketidakpastian dan variasi inilah yang membuat pengaturan statis tidak memadai, karena parameter yang optimal pada satu kondisi dapat menjadi suboptimal atau bahkan kontraproduktif pada kondisi lainnya. Oleh karena itu, diperlukan model yang mampu menangkap hubungan dinamis antara state dan kontrol sehingga sistem dapat diatur secara adaptif sesuai perubahan kondisi.

Analisis ini menunjukkan bahwa inti permasalahan bukan sekadar menentukan nilai batch atau interval polling yang paling cepat atau paling efisien, tetapi menemukan cara untuk menjaga keseimbangan antara backlog, latensi, dan utilisasi sumber daya dalam kondisi sistem yang terus berubah. Model pengendalian dibutuhkan untuk menjaga agar variabel state bergerak menuju kondisi yang stabil dan terkendali, terlepas dari variasi beban. Hasil analisis inilah yang menjadi dasar untuk merancang pendekatan pengendalian yang akan dibahas pada subbab berikutnya.

III.2. Rancangan Solusi

Berdasarkan analisis pada bagian sebelumnya, permasalahan sinkronisasi data pada arsitektur CQRS menunjukkan karakteristik sistem dinamis yang dipengaruhi oleh interaksi antara kondisi sistem dan parameter pengaturan pemrosesan. Backlog, latensi sinkronisasi, dan utilisasi sumber daya tidak berkembang secara independen, melainkan saling mempengaruhi seiring perubahan beban dan konfigurasi sinkronisasi. Oleh karena itu, solusi yang dirancang dalam penelitian ini tidak bertujuan mencari konfigurasi parameter statis yang optimal, melainkan mekanisme pengendalian yang mampu

menyesuaikan parameter sinkronisasi secara adaptif agar sistem tetap berada pada kondisi seimbang.

Pendekatan pengendalian yang diusulkan dirancang dengan memformulasikan proses sinkronisasi sebagai sistem dinamis dalam kerangka *state-space*. Dengan formulasi ini, hubungan antara kondisi sistem (*state*) dan parameter yang dapat dikendalikan (*control input*) dapat dianalisis dan diatur secara sistematis. Pendekatan ini memungkinkan penerapan berbagai strategi pengendalian, mulai dari kontrol umpan balik konvensional hingga kontrol optimal dan pendekatan berbasis data, dalam kerangka yang konsisten.

III.2.1. Abstraksi Sistem dan Pemodelan State-Space

Untuk merancang mekanisme pengendalian, proses sinkronisasi CQRS diabstraksikan sebagai sistem dinamis diskrit yang berevolusi seiring waktu. Abstraksi ini memandang pipeline sinkronisasi, mulai dari konsumsi event hingga materialisasi *read model*, sebagai sebuah sistem yang memiliki keadaan internal dan dapat dipengaruhi melalui aksi kontrol tertentu. Pendekatan ini sejalan dengan praktik pemodelan sistem komputasi sebagai sistem dinamis ketika perilaku sistem dipengaruhi oleh umpan balik dan perubahan beban (Ogata, 2010).

Pada Message Sink terdapat Persamaan II.1 nilai-nilai matriks A, B, C, dan D didapatkan dari ekperimen dengan pendekatan open loop agar diketahui pengaruh variable-variable kontrol terhadap variable-variable state (u). Dengan vektor aksi kontrol $\underline{u}$ terdefinisi sebagai

$$u = \begin{bmatrix} b \\ p \end{bmatrix} \tag{III.1}$$

1. b adalah *batch size,* yaitu jumlah pesan yang diproses dalam satu batch
2. p adalah *poll interval*, yaitu waktu yang jeda yang ditentukan untuk sistem dapat memroses batch data.

Vektor state x terdefinisi sebagai

$$x = \begin{bmatrix} n \\ c \\ m \\ o \end{bmatrix} \tag{III.2}$$

1. *n* adalah banyaknya pesan pada event source,
2. *c* adalah *cpu utilization,*
3. *m* adalah *memory utilization*
4. *o* adalah *I/O operation*

Pemilihan variabel ini didasarkan pada dua pertimbangan utama. Pertimbangan tersebut adalah variabel dapat diobservasi atau diukur secara real-time dan juga variabel tersebut memiliki pengaruh langsung terhadap laju sinkronisasi dan tekanan terhadap *read database*

Dalam konteks sistem nyata, hubungan antara *state* dan *control* bersifat nonlinier. Namun, untuk keperluan perancangan kontrol optimal, sistem diasumsikan dapat didekati secara linier pada rentang operasi tertentu (*linearization around operating point*). Selain itu, diasumsikan bahwa perubahan beban dalam satu interval kontrol relatif kecil (*quasi-stationary assumption*), sehingga pendekatan linier diskrit masih valid untuk perancangan kendali (Ogata, 2010; Hellerstein et al., 2004). Validitas asumsi ini kemudian diuji melalui eksperimen *open-loop* pada tahap persiapan penelitian.

III.2.2.Perancangan Fungsi Biaya dan Definisi Keseimbangan

Tujuan pengendalian dalam penelitian ini adalah menjaga keseimbangan antara kecepatan sinkronisasi data dan efisiensi penggunaan sumber daya. Keseimbangan tersebut tidak dapat direpresentasikan oleh satu metrik tunggal, melainkan merupakan kompromi antara beberapa tujuan yang saling bertentangan, seperti minimisasi backlog, pengurangan latensi sinkronisasi, dan pembatasan utilisasi sumber daya agar tidak melampaui kapasitas sistem (Bailis et al., 2012; Wu et al., 2020).

Untuk merepresentasikan tujuan tersebut secara formal, digunakan fungsi biaya kuadratik yang didefinisikan pada persamaan II.2

Komponen pertama dari fungsi biaya memberikan penalti terhadap deviasi kondisi sistem dari keadaan yang diinginkan, sementara komponen kedua memberikan penalti terhadap besarnya aksi kontrol yang diterapkan. Dengan formulasi ini, sistem tidak hanya diarahkan untuk mengurangi backlog atau utilisasi sumber daya, tetapi juga menghindari perubahan parameter sinkronisasi yang terlalu agresif atau terlalu lambat.

Makna optimal dalam kerangka LQR didefinisikan sebagai kebijakan kontrol yang meminimalkan fungsi biaya tersebut sepanjang horizon waktu, sehingga menghasilkan kompromi terbaik antara perbaikan kondisi sistem dan besarnya perubahan parameter sinkronisasi. Bobot pada matriks Q mencerminkan tingkat kepentingan relatif masing-masing variabel state, misalnya penekanan yang lebih besar pada backlog akan mendorong sistem untuk mempercepat sinkronisasi, sementara penekanan pada utilisasi sumber daya akan menghasilkan perilaku pengendalian yang lebih konservatif. Matriks R berfungsi untuk membatasi perubahan *batch size* dan *poll interval* agar tidak terlalu agresif, sehingga mengurangi potensi osilasi dan ketidakstabilan sistem.

Pada pendekatan PID dan ANN, fungsi biaya kuadratik tidak digunakan secara langsung sebagai dasar perhitungan aksi kontrol. PID menghasilkan aksi kontrol berdasarkan kesalahan dan dinamika error sistem, sedangkan ANN mempelajari pemetaan antara kondisi sistem dan aksi kontrol dari data. Namun demikian, tujuan pengendalian yang sama tetap digunakan sebagai dasar evaluasi performa, sehingga perbandingan antar metode dapat dilakukan secara konsisten. Dalam konteks ANN, fungsi biaya LQR dan variasi bobotnya juga dapat dimanfaatkan sebagai acuan untuk membentuk data pelatihan yang merepresentasikan aksi kontrol yang diinginkan pada kondisi sistem tertentu, tanpa menjadikan fungsi biaya tersebut sebagai bagian dari mekanisme inferensi ANN secara langsung.

Dengan pemisahan yang jelas antara tujuan pengendalian umum dan formulasi fungsi biaya khusus LQR, perancangan solusi pengendalian pada penelitian ini tetap konsisten secara teoritis dan memungkinkan evaluasi yang adil antara pendekatan berbasis model dan pendekatan berbasis data.

III.2.3. Kontrol Optimal Linear Quadratic Regulator (LQR)

Kontrol optimal LQR digunakan sebagai pendekatan berbasis model yang secara eksplisit memformulasikan tujuan pengendalian dalam bentuk fungsi biaya. LQR bekerja berdasarkan representasi *state-space* sistem, di mana kondisi sinkronisasi direpresentasikan sebagai vektor state dan parameter sinkronisasi sebagai vektor kontrol. Dengan model ini, LQR menghitung kebijakan kontrol yang meminimalkan fungsi biaya

kuadratik yang merepresentasikan keseimbangan antara perbaikan kondisi sistem dan besarnya aksi kontrol.pen

Berbeda dengan PID, LQR mempertimbangkan seluruh variabel state secara simultan dan tidak hanya berfokus pada satu sinyal kesalahan. Pendekatan ini memungkinkan pengendalian yang lebih sistematis terhadap trade-off antara backlog, latensi sinkronisasi, dan utilisasi sumber daya. Kebijakan kontrol LQR diperoleh melalui penyelesaian persamaan Riccati, sehingga aksi kontrol yang dihasilkan bersifat optimal secara matematis pada asumsi linearitas lokal dan gangguan terbatas.

Keterbatasan utama LQR terletak pada kebutuhan akan model sistem dan penentuan bobot fungsi biaya yang tepat. Proses identifikasi model dan penalaan bobot memerlukan eksperimen awal dan pemahaman terhadap dinamika sistem. Meskipun demikian, LQR memberikan kerangka pengendalian yang interpretable dan konsisten secara teori, sehingga cocok digunakan sebagai pendekatan utama dalam penelitian ini.

III.3. Rancangan Solusi

Tahap persiapan penelitian dilakukan untuk memperoleh data yang merepresentasikan dinamika sinkronisasi sistem pada berbagai kondisi beban. Data ini digunakan untuk identifikasi hubungan antara kondisi sistem dan parameter sinkronisasi, serta sebagai dasar pelatihan model pembelajaran berbasis data. Seluruh proses persiapan dirancang agar konsisten dengan tujuan pengendalian dan metrik evaluasi yang digunakan pada tahap pengujian.

III.3.1. Eksperimen Open-Loop dan Identifikasi Sistem

Eksperimen open-loop dilakukan dengan menjalankan sistem sinkronisasi tanpa mekanisme pengendalian adaptif aktif. Parameter sinkronisasi diatur secara manual dan divariasikan dalam rentang yang telah ditentukan, sementara kondisi sistem diamati dan dicatat secara periodik. Pendekatan ini dipilih untuk memastikan bahwa hubungan sebab akibat antara parameter sinkronisasi dan respons sistem dapat diamati secara langsung, tanpa dipengaruhi oleh umpan balik pengendali.

Pada setiap interval pengamatan, kondisi sistem dicatat sebagai vektor state yang mencakup backlog event, utilisasi CPU, utilisasi memori, dan intensitas operasi I/O pada read database. Secara bersamaan, parameter kontrol berupa batch size dan poll interval dicatat sebagai bagian dari konfigurasi eksperimen. Pengukuran dilakukan pada interval waktu yang tetap untuk menjaga konsistensi temporal antar variabel.

Eksperimen dijalankan pada beberapa skenario beban yang berbeda, termasuk beban rendah, sedang, dan tinggi. Untuk setiap skenario, beberapa kombinasi parameter sinkronisasi diuji dan dijalankan selama jendela waktu observasi yang sama. Selama periode tersebut, metrik performa seperti perubahan backlog, latensi sinkronisasi, dan utilisasi sumber daya dicatat untuk dianalisis lebih lanjut. Data tersebut dimanfaatkan untuk mengamati karakteristik dinamika sistem dan memvalidasi asumsi pemodelan yang digunakan pada perancangan kontrol.

III.3.2. Definisi Aksi Kontrol Optimal

Aksi kontrol optimal $u^*(t)$ didefinisikan sebagai parameter sinkronisasi yang menghasilkan kinerja terbaik untuk suatu kondisi sistem tertentu berdasarkan tujuan pengendalian yang telah ditetapkan. Dalam penelitian ini, optimalitas tidak diartikan sebagai minimisasi satu metrik secara terpisah, melainkan sebagai pencapaian keseimbangan antara backlog, latensi sinkronisasi, dan pemanfaatan sumber daya dalam suatu horizon pengamatan yang terbatas.

Secara formal, untuk kondisi sistem x(t), aksi kontrol optimal didefinisikan sebagai

$$u^*(t) = \min_{u \in U} J(x(t),\ u) \tag{III.1}$$

di mana U adalah himpunan kandidat aksi kontrol yang memenuhi batas operasional sistem, dan J adalah fungsi biaya yang merepresentasikan tujuan pengendalian. Fungsi biaya dirumuskan pada dengan rumus II.2

Pada suatu kondisi awal sistem $u^*(t)$ yang diamati, beberapa kombinasi parameter sinkronisasi, yaitu batch size dan poll interval, diuji secara terpisah. Selama satu periode pengamatan, aksi kontrol dijaga tetap konstan untuk setiap kombinasi yang diuji, sementara respons sistem dicatat secara periodik. Nilai fungsi biaya kemudian dihitung dari respons

tersebut, dan kombinasi parameter yang menghasilkan nilai fungsi biaya minimum ditetapkan sebagai aksi kontrol optimal untuk kondisi sistem tersebut.

Pendekatan ini memastikan bahwa definisi $u^*(t)$ konsisten dengan tujuan pengendalian LQR, meskipun diperoleh melalui evaluasi empiris. Dengan demikian, $u^*(t)$ merepresentasikan aksi kontrol yang paling seimbang untuk kondisi sistem tertentu dalam batas asumsi dan horizon pengamatan yang digunakan, serta dapat digunakan sebagai acuan dalam analisis performa dan perbandingan pada tahap evaluasi.

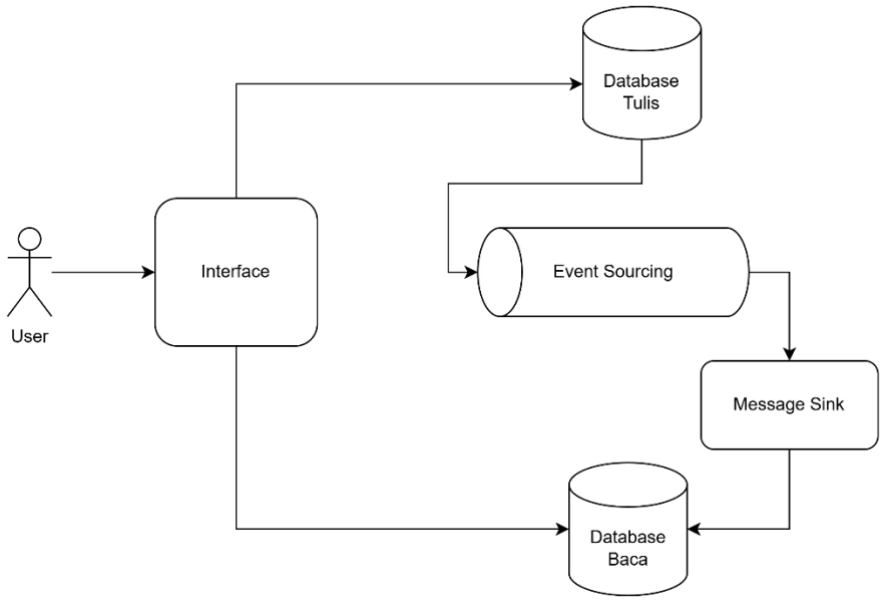
III.4. Desain eksperimen dan skenarion pengujian

Desain eksperimen pada penelitian ini bertujuan untuk mengevaluasi efektivitas pengendalian sinkronisasi data berbasis LQR dalam berbagai kondisi operasional sistem. Eksperimen dirancang secara terkontrol agar hasil yang diperoleh mencerminkan pengaruh langsung dari mekanisme pengendalian terhadap dinamika sinkronisasi, bukan akibat perubahan lingkungan atau konfigurasi sistem yang tidak relevan.

Eksperimen dilakukan dengan menjalankan sistem CQRS pada lingkungan uji yang konsisten, dengan konfigurasi perangkat keras, skema data, dan logika materialisasi read model yang dijaga tetap. Variasi yang diperkenalkan dalam eksperimen dibatasi pada skenario beban dan parameter pengendalian yang menjadi fokus penelitian. Pendekatan ini memastikan validitas internal eksperimen dan memungkinkan interpretasi hasil yang lebih jelas.

III.4.1. Lingkungan Uji

Lingkungan uji dirancang untuk merepresentasikan kondisi operasional sistem sinkronisasi data berbasis CQRS secara realistis dan terkontrol. Seluruh eksperimen dijalankan pada arsitektur yang sama untuk memastikan bahwa perbedaan kinerja yang diamati berasal dari mekanisme pengendalian, bukan dari perubahan konfigurasi sistem atau lingkungan eksekusi.



Gambar III.2 Arsitektur Desain yang akan diuji

Arsitektur sistem uji terdiri dari empat komponen utama, yaitu write model sebagai sumber perubahan data, message broker sebagai media distribusi event, message sink sebagai konsumer event sekaligus titik penerapan kontrol, dan read database sebagai target materialisasi data. Alur sinkronisasi dimulai dari operasi tulis pada write model yang menghasilkan event perubahan data, kemudian event tersebut dipublikasikan ke message broker dan dikonsumsi oleh message sink untuk diproses secara bertahap sebelum diterapkan ke read database.

Mekanisme kontrol LQR ditempatkan pada message sink karena komponen ini memiliki peran langsung dalam menentukan laju konsumsi event dan intensitas beban yang diberikan ke read database. Parameter sinkronisasi yang dikendalikan meliputi batch size dan poll interval, yang secara langsung memengaruhi jumlah event yang diproses dalam satu siklus serta frekuensi pemrosesan. Dengan menempatkan kontrol pada message sink, sistem memperoleh fleksibilitas untuk menyesuaikan laju sinkronisasi tanpa mengganggu logika bisnis pada write model maupun pola akses pada sisi baca.

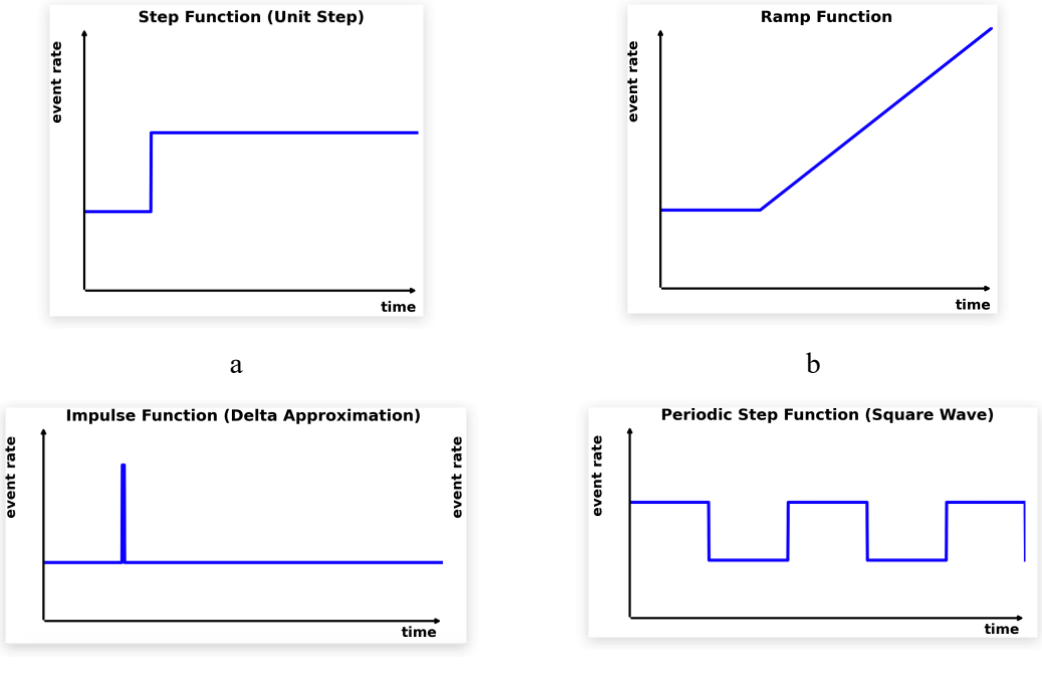
Selama eksperimen, konfigurasi perangkat keras dan perangkat lunak dijaga tetap. Kapasitas CPU, memori, dan penyimpanan tidak diubah, demikian pula versi sistem operasi, message broker, dan basis data yang digunakan. Skema data pada write model dan

read database serta logika materialisasi data juga tidak mengalami perubahan. Pendekatan ini dilakukan untuk menjaga validitas internal eksperimen dan memastikan bahwa dinamika sistem yang diamati mencerminkan pengaruh pengendalian terhadap proses sinkronisasi.

Sebelum setiap eksperimen dimulai, sistem dijalankan hingga mencapai kondisi awal yang stabil dengan konfigurasi parameter sinkronisasi default. Periode stabilisasi ini bertujuan untuk menghindari pengaruh transien awal sistem terhadap hasil pengukuran. Setelah kondisi awal tercapai, mekanisme kontrol LQR diaktifkan dan eksperimen dijalankan sesuai dengan skenario beban yang telah ditentukan.

III.4.2. Perancangan Skenario Beban

Skenario beban dirancang untuk mengevaluasi kinerja pengendalian sinkronisasi data berbasis LQR pada berbagai kondisi operasional dan pola perubahan beban. Selain variasi tingkat beban, penelitian ini juga mempertimbangkan karakteristik temporal perubahan beban karena respons sistem kendali sangat dipengaruhi oleh bagaimana beban berubah terhadap waktu.



Gambar III.3 Simulasi beban dengan beberapa skenario fungi (a) fungsi *step*, (b) fungsi *ramp*, (c) fungsi *impulse* dan (d) fungsi *step* yang berulang

Skenario beban dengan pola *step* digunakan untuk merepresentasikan perubahan beban yang terjadi secara tiba-tiba, misalnya lonjakan laju produksi event akibat peningkatan trafik secara mendadak. Pola ini digunakan untuk mengamati respons transien sistem, khususnya waktu respons awal, kenaikan backlog, dan kemampuan pengendalian dalam meredam lonjakan beban secara cepat.

Pola *ramp* digunakan untuk merepresentasikan perubahan beban yang meningkat atau menurun secara bertahap. Pola ini mencerminkan kondisi pertumbuhan trafik yang progresif dan memungkinkan evaluasi kemampuan pengendalian dalam menyesuaikan parameter sinkronisasi secara gradual tanpa menimbulkan osilasi yang berlebihan.

Pola *impulse* digunakan untuk merepresentasikan lonjakan beban sesaat dengan durasi singkat (*short burst*), yang kemudian diikuti oleh kembalinya beban ke kondisi awal. Pola ini bertujuan untuk mengamati sensitivitas sistem terhadap gangguan sementara dan kemampuan pengendalian dalam memulihkan kondisi sistem setelah gangguan terjadi.

Selain itu, digunakan pola *step periodik* untuk merepresentasikan perubahan beban yang berulang antara dua tingkat tertentu dalam interval waktu tetap. Pola ini digunakan untuk mengevaluasi stabilitas jangka menengah sistem pengendalian serta potensi munculnya osilasi akibat perubahan beban yang berulang. Pengamatan pada skenario ini difokuskan pada konsistensi respons sistem dan kemampuan pengendalian dalam mempertahankan performa yang stabil pada siklus beban yang berulang.

Dengan menggunakan kombinasi keempat pola beban tersebut, eksperimen diharapkan mampu memberikan gambaran yang komprehensif mengenai respons dinamis dan stabilitas pengendalian sinkronisasi pada berbagai kondisi operasional yang relevan.

III.5. Metrik Evaluasi Kerja

Evaluasi kinerja dalam penelitian ini dilakukan untuk menilai sejauh mana mekanisme pengendalian berbasis Linear Quadratic Regulator (LQR) mampu menjaga keseimbangan antara kualitas sinkronisasi data dan efisiensi penggunaan sumber daya sistem. Oleh karena itu, metrik evaluasi dipilih secara selektif agar dapat merepresentasikan tujuan pengendalian yang telah dirumuskan pada tahap perancangan solusi, serta mampu menangkap respons sistem pada berbagai kondisi beban dan pola perubahan beban.

Metrik evaluasi dikelompokkan berdasarkan perannya dalam analisis kinerja sistem, meliputi metrik inti kinerja sinkronisasi, metrik dinamika kontrol, metrik efisiensi pemrosesan, dan metrik biaya kontrol. Pengelompokan ini bertujuan untuk memisahkan pengukuran kualitas sinkronisasi, karakteristik respons dinamis, efisiensi operasional, dan kesesuaian solusi terhadap formulasi optimasi yang digunakan. Seluruh metrik diukur secara periodik selama eksperimen dan digunakan secara konsisten sebagai dasar analisis dan perbandingan pada bab hasil dan pembahasan.

III.5.1. Metrik Inti Kinerja Sinkronisasi

Metrik inti kinerja sinkronisasi digunakan untuk mengevaluasi kualitas dasar proses sinkronisasi data pada arsitektur CQRS. Metrik ini dipilih karena secara langsung merepresentasikan tujuan utama pengendalian, yaitu menjaga konsistensi data dan kestabilan sistem pada berbagai kondisi beban. Seluruh metrik inti diukur secara periodik selama eksperimen dan dianalisis baik pada kondisi tunak maupun transien.

Backlog sinkronisasi digunakan sebagai indikator utama untuk mengukur tingkat ketertinggalan proses sinkronisasi. Backlog didefinisikan sebagai jumlah event yang telah dihasilkan oleh write model tetapi belum diproses oleh message sink pada suatu waktu tertentu. Nilai backlog yang tinggi menunjukkan ketidakseimbangan antara laju produksi dan konsumsi event, sedangkan pengendalian yang efektif ditunjukkan oleh kemampuan menekan pertumbuhan backlog dan mempercepat pemulihannya setelah terjadi gangguan. Backlog diukur dalam satuan jumlah event, dengan analisis mencakup nilai rata rata, nilai maksimum, dan waktu pemulihan menuju kondisi referensi.

Latensi sinkronisasi digunakan untuk mengukur keterlambatan propagasi perubahan data dari write model hingga perubahan tersebut tersedia pada read database. Latensi ini mencerminkan konsistensi data yang dapat diamati oleh pengguna dan berhubungan langsung dengan kualitas layanan sistem. Pengukuran latensi dilakukan dengan mencatat selisih waktu antara timestamp event pada sisi tulis dan waktu penerapan perubahan pada sisi baca. Satuan yang digunakan adalah milidetik atau detik, dengan analisis meliputi nilai rata rata dan nilai maksimum selama periode eksperimen.

Utilisasi sumber daya digunakan untuk mengevaluasi dampak pengendalian terhadap kapasitas sistem, khususnya pada komponen read database yang menjadi target materialisasi data. Metrik ini mencakup utilisasi CPU, penggunaan memori, dan aktivitas I/O selama proses sinkronisasi berlangsung. Utilisasi CPU dan memori diukur dalam persen, sedangkan aktivitas I/O diukur dalam operasi per detik atau jumlah data yang diproses per satuan waktu. Metrik ini digunakan untuk memastikan bahwa pengendalian mampu menjaga keseimbangan antara peningkatan laju sinkronisasi dan keterbatasan sumber daya sistem.

No	Metrik	Definisi Operasional	Satuan
1	Backlog Sinkronisasi	Jumlah event yang telah dihasilkan oleh write model tetapi belum diproses oleh message sink pada waktu tertentu	Event
2	Backlog Rata Rata	Nilai rata rata backlog selama satu periode eksperimen	Event
3	Backlog Maksimum	Nilai backlog tertinggi yang tercatat selama eksperimen	Event
4	Waktu Pemulihan Backlog	Waktu yang dibutuhkan backlog untuk kembali mendekati nilai referensi setelah terjadi gangguan beban	Detik
5	Latensi Sinkronisasi Rata Rata	Rata rata waktu dari event dihasilkan hingga diterapkan pada read database	Milidetik atau detik
6	Latensi Sinkronisasi Maksimum	Nilai latensi sinkronisasi tertinggi selama eksperimen	Milidetik atau detik
7	Utilisasi CPU	Persentase penggunaan CPU pada read database selama proses sinkronisasi	Persen
8	Utilisasi Memori	Persentase penggunaan memori pada read database selama proses sinkronisasi	Persen
9	Aktivitas I O	Intensitas operasi baca tulis selama proses sinkronisasi	Persen

Tabel III.1 Metrik kinerja sinkronisasi

III.5.2. Metrik Dinamika Kontrol

Metrik dinamika kontrol digunakan untuk mengevaluasi karakteristik respons sistem terhadap perubahan beban, khususnya pada kondisi transien seperti pola step, ramp, impulse, dan step periodik. Berbeda dengan metrik inti yang menilai kualitas sinkronisasi secara umum, metrik dinamika kontrol berfokus pada bagaimana sistem bereaksi terhadap gangguan dan seberapa cepat sistem kembali ke kondisi yang diinginkan. Metrik ini sangat relevan dalam konteks pengendalian berbasis LQR karena secara langsung mencerminkan kualitas respons kendali yang dihasilkan.

Evaluasi dinamika kontrol dilakukan dengan mengamati perilaku temporal backlog sebagai variabel state utama. Respons backlog terhadap perubahan beban digunakan untuk menilai kecepatan respons, tingkat agresivitas pengendalian, dan kestabilan menuju kondisi referensi. Dengan menggunakan metrik dinamika kontrol, penelitian ini dapat membedakan pengendalian yang cepat namun berisiko agresif dari pengendalian yang lebih stabil dan terkontrol.

No	Metrik	Definisi Operasional	Satuan
1	Rise Time	Waktu yang dibutuhkan backlog untuk kembali mendekati nilai referensi setelah terjadi perubahan beban	Detik
2	Overshoot	Selisih maksimum backlog yang melebihi nilai referensi setelah terjadi perubahan beban	Persen
3	Settling Time	Waktu yang dibutuhkan hingga backlog berada dan bertahan dalam rentang toleransi tertentu di sekitar nilai referensi	Detik

Tabel III.2 Metrik dinamika kontrol

Rise time digunakan untuk mengukur kecepatan respons pengendalian terhadap gangguan beban. Nilai rise time yang kecil menunjukkan bahwa mekanisme kontrol mampu merespons perubahan kondisi sistem secara cepat. Namun, respons yang terlalu cepat dapat meningkatkan risiko overshoot, sehingga rise time perlu dianalisis bersamaan dengan metrik lainnya.

Overshoot merepresentasikan tingkat agresivitas pengendalian setelah terjadi perubahan beban. Overshoot yang besar menunjukkan bahwa pengendalian memberikan respons berlebihan, yang berpotensi meningkatkan tekanan terhadap sistem. Dalam konteks sinkronisasi data, overshoot backlog yang tinggi dapat mengindikasikan risiko overload sementara pada read database.

Settling time digunakan untuk mengevaluasi kestabilan respons pengendalian. Metrik ini menunjukkan berapa lama sistem membutuhkan waktu untuk kembali dan bertahan pada kondisi yang mendekati nilai referensi setelah gangguan terjadi. Settling time yang singkat tanpa overshoot berlebihan menunjukkan karakteristik pengendalian yang stabil dan efektif.

III.5.3. Metrik Efisiensi Pemrosesan

Metrik efisiensi pemrosesan digunakan untuk mengevaluasi sejauh mana mekanisme pengendalian sinkronisasi mampu meningkatkan kinerja pemrosesan event tanpa mengorbankan efisiensi penggunaan sumber daya. Dalam konteks sistem CQRS, pengendalian yang efektif tidak hanya ditunjukkan oleh penurunan backlog atau latensi, tetapi juga oleh kemampuan sistem dalam memproses event secara efisien dengan sumber daya yang tersedia.

Evaluasi efisiensi pemrosesan penting karena pengendalian yang terlalu agresif dapat meningkatkan throughput dalam jangka pendek, tetapi berpotensi menimbulkan pemborosan sumber daya atau tekanan berlebihan pada sistem. Oleh karena itu, metrik efisiensi digunakan untuk menilai keseimbangan antara kapasitas pemrosesan yang dicapai dan biaya sumber daya yang dikeluarkan selama proses sinkronisasi berlangsung.

No	Metrik	Definisi Operasional	Satuan
1	Throughput Pemrosesan	Jumlah event yang berhasil diproses oleh message sink per satuan waktu	Event per detik
2	Efisiensi CPU	Rasio antara throughput pemrosesan dan utilisasi CPU	Event per detik per persen CPU
3	Efisiensi Memori	Rasio antara throughput pemrosesan dan utilisasi memori	Event per detik per persen memori

Tabel III.3 Metrik efisiensi pemrosesan

Throughput pemrosesan digunakan untuk mengukur kapasitas sistem dalam memproses event selama proses sinkronisasi. Nilai throughput yang tinggi menunjukkan bahwa sistem mampu menangani laju event yang besar, namun metrik ini perlu dianalisis bersamaan dengan utilisasi sumber daya agar tidak menimbulkan interpretasi yang menyesatkan.

Efisiensi CPU dan efisiensi memori digunakan untuk mengevaluasi seberapa efektif sumber daya sistem dimanfaatkan dalam mendukung proses sinkronisasi. Nilai efisiensi yang tinggi menunjukkan bahwa peningkatan throughput dicapai dengan penggunaan sumber daya yang relatif efisien. Dengan mengombinasikan metrik throughput dan efisiensi sumber daya, evaluasi kinerja tidak hanya berfokus pada kecepatan pemrosesan, tetapi juga pada keberlanjutan operasional sistem.

III.5.4. Metrik Biaya Kontrol

Metrik biaya kontrol digunakan untuk mengevaluasi kinerja pengendalian dari sudut pandang optimasi yang mendasari perancangan Linear Quadratic Regulator (LQR). Berbeda dengan metrik kinerja dan efisiensi yang menilai dampak pengendalian pada sistem secara eksternal, metrik biaya kontrol menilai sejauh mana kebijakan kontrol yang diterapkan selaras dengan tujuan optimasi yang telah diformulasikan melalui fungsi biaya.

Evaluasi biaya kontrol penting untuk memastikan bahwa perbaikan kinerja sinkronisasi tidak dicapai melalui aksi kontrol yang berlebihan. Dalam konteks ini, biaya kontrol digunakan untuk mengukur keseimbangan antara perbaikan kondisi sistem dan besarnya aksi kontrol yang diterapkan. Dengan menganalisis nilai biaya kontrol, penelitian ini dapat menilai konsistensi perilaku pengendalian LQR terhadap tujuan optimasi yang dirancang

No	Metrik	Definisi Operasional	Satuan
1	Nilai Fungsi Biaya	Akumulasi penalti state dan aksi kontrol selama horizon pengamatan sesuai fungsi biaya kuadratik LQR	Tidak berdimensi
2	Biaya State	Komponen fungsi biaya yang merepresentasikan penalti terhadap deviasi state dari nilai referensi	Tidak berdimensi
3	Biaya Aksi Kontrol	Komponen fungsi biaya yang merepresentasikan penalti terhadap besarnya aksi kontrol yang diterapkan	Tidak berdimensi

Tabel III.4 Metrik biaya kontrol

Nilai fungsi biaya digunakan sebagai indikator utama untuk mengevaluasi performa pengendalian LQR dari perspektif optimasi. Nilai fungsi biaya yang lebih rendah menunjukkan bahwa sistem berada lebih dekat dengan kondisi yang diinginkan dengan besaran aksi kontrol yang relatif terkendali.

Biaya state digunakan untuk menilai sejauh mana kondisi sistem menyimpang dari nilai referensi selama proses sinkronisasi. Nilai biaya state yang tinggi mengindikasikan bahwa sistem sering berada jauh dari kondisi target, misalnya akibat backlog yang besar atau utilisasi sumber daya yang berlebihan.

Biaya aksi kontrol digunakan untuk mengevaluasi tingkat agresivitas pengendalian. Nilai biaya aksi kontrol yang tinggi menunjukkan bahwa perbaikan kondisi sistem dicapai melalui perubahan parameter sinkronisasi yang besar atau sering. Dengan menganalisis

kedua komponen biaya secara terpisah, penelitian ini dapat mengevaluasi keseimbangan antara kualitas sinkronisasi dan intensitas aksi kontrol yang diterapkan.

DAFTAR PUSTAKA

CQRS, data diperoleh melalui situs internet: https://martinfowler.com/bliki/CQRS.html. Diunduh pada tanggal 6 Juni 2025.

Vogels, W. (2009). Eventually Consistent. *Communications of the ACM*, 40–44.

Kindsoon, Munoye., Martinek, Péter (2023): A Simplified Approach to Distributed Message Handling in a CQRS Architecture, *Acta Polytechnica Hungarica.*

Armag˘an, Özgür., Gören-Sümer, Leyla (2014): Feedback control for multi-resource usage of virtualized database server, *Computers and Electrical Engineering.*

Pan, Wenping., Mu, Dejun., Wu, Hangxing., Yao, Lei (2008): Feedback Control-based QoS Guarantees in Web Application Servers, *IEEE International Conference on High Performance Computing and Communications.*

Gong, Siqian., Yin, Beibei., Cai, Kai-yuan (2018): An Adaptive PID Control for QoS Management in Cloud Computing System, *IEEE International Symposium on Software Reliability Engineering Workshops.*

Wu, Han., Shang, Zhiao., Guang, Peng., Wolter, Katinka (2020). A Reactive Batching Strategy of Apache Kafka for Reliable Stream Processing in Real-time, *IEEE International Symposium on Software Reliability Engineering.*

ZangLong (2017). Improvement and Implementation of a High Performance CQRS Architecture, *International Conference on Robots & Intelligent System..*

Bailis, Peter., Venkataraman, Shivaram., Franklin, Michael J., Hellerstein, Joseph M., Stoica, Ion (2017). Probabilistically Bounded Staleness for Practical Partial Quorums, *International Conference on Robots & Intelligent System.*

Kindsoon, Munoye., Martinek, Péter (2020). Evaluation of Data Storage Patterns in Microservices Archicture, *IEEE 15th International Conference of System of Systems Engineering*

Ogata, K. (2010). *Modern Control Engineering* (5th ed.). Prentice Hall. 793–798

Kleppmann, M. (2017). *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. O'Reilly Media. 389-391

Newman, S. (2015). *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media. 193–198

Brewer, E. (2000). Towards Robust Distributed Systems. *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing (PODC)*.