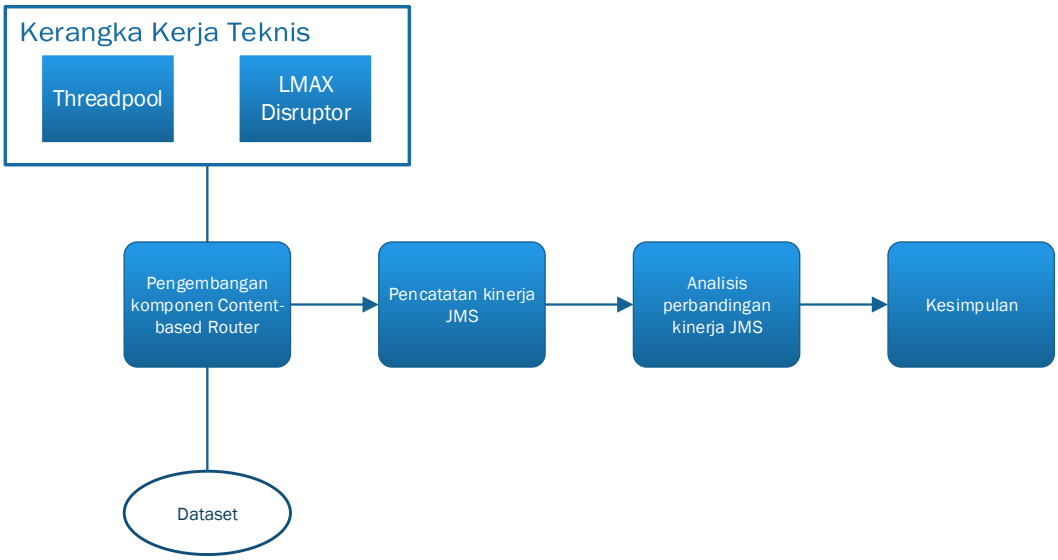
BAB III

METODOLOGI PENELITIAN

1.1 Kerangka Penelitian

Berikut adalah kerangka kerja teknis dalam penelitian ini,



Gambar 3.1 Kerangka Penelitian

Kerangka penelitian pada gambar 2.13 dapat dijelaskan sebagai berikut:

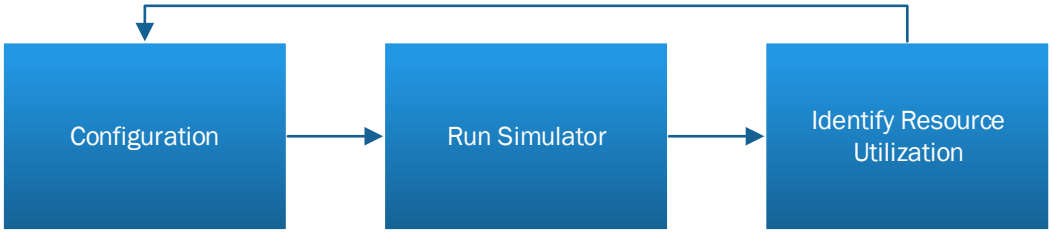
1) Pengembangan komponen *content-based router*, menghasilkan sebuah aplikasi yang berperan sebagai *router* di dalam sebuah sistem MoM. Aplikasi ini mendukung pemrosesan *message* berdasarkan kerangka kerja teknis Threadpool atau LMAX Disruptor.

2) Pengembangan komponen *message simulator*, menghasilkan sebuah aplikasi yang mewakili sistem eksternal yang terhubung dalam aplikasi MoM serta bertugas mengirimkan *message* (dataset) secara konsisten kepada *router*. *Message* yang dikirimkan memiliki karakteristik format tertentu dengan nilai yang bervariasi.
3) Pencatatan kinerja dilakukan dengan mengambil nilai dari variabel – variabel berjalan yang dibutuhkan untuk analisis kelak, yaitu *elapsed time*, penggunaan memory dan cpu. Pencatatan akan dilakukan langsung dalam periode waktu tertentu oleh aplikasi router dengan menggunakan *file* log.
4) Analisis perbandingan kinerja dilakukan berdasarkan data kinerja yang didapat pada tahap 3. Tahap ini dilakukan dalam kurun waktu tertentu, dan dilakukan pada simulasi dengan *router* yang memiliki kerangka kerja teknis berbeda, yaitu menggunakan Threadpool atau LMAX Disruptor.
5) Kesimpulan akhir dilakukan dengan mencari tipe *router* dengan kerangka kerja teknis seperti apakah yang memiliki kinerja paling baik dalam jangka waktu dan konfigurasi tertentu, dengan memperhatikan hubungan antara buffer size, *message* yang di-feed dan penggunaan resources. Kesimpulan yang didapat akan menentukan apakah kombinasi Threadpool dan LMAX Disruptor layak dilakukan pada kondisi tertentu atau cukup dengan implementasi salah satu saja.

1.2 Desain Penelitian

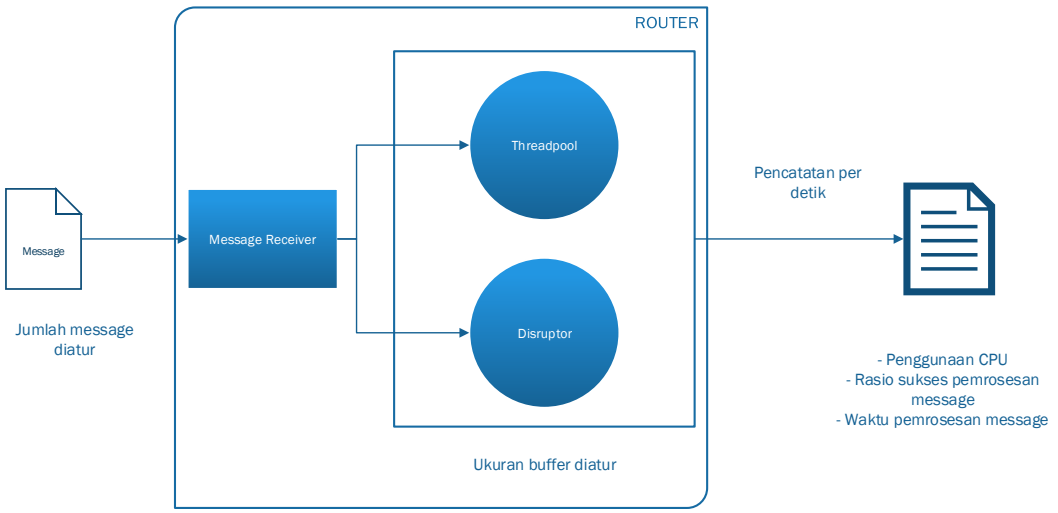
Penelitian ini menggunakan metode kuantitatif, karena cocok dengan kerangka kerja penelitian yang membutuhkan analisis dan perhitungan matematis untuk data yang berjumlah banyak dan memiliki karakteristik format tertentu. Dalam penelitian ini akan dibangun sebuah sistem yang menyerupai sistem dunia nyata yang telah berjalan dan variabel operasional yang dibutuhkan akan diambil berdasar interval waktu tertentu untuk kemudian dilakukan analisis dari data yang telah didapat. Penelitian ini terletak pada pencarian kinerja yang optimal di antara dua buah kerangka kerja teknis *inter-threading* yaitu LMAX Disruptor dan Threadpool, masing – masing mengklaim memiliki kinerja yang baik berdasarkan konfigurasi dan kasus tertentu. Dengan memilih batasan pada kasus spesifik yaitu *middleware* berorientasi *message*, pertimbangan – pertimbangan dan konfigurasi tertentu dapat diterapkan dan hasil yang didapat diukur berdasarkan ketentuan standar dari pengukuran kinerja JMS.

Seperti yang ditunjukkan pada gambar 3.1, pertama akan dilakukan konfigurasi dahulu, termasuk setup *framework* yang dipakai *router* menggunakan konfigurasi *default*. Simulator akan dijalankan untuk mengirimkan sejumlah *message* dan sistem akan mencatat penggunaan *resources* per satuan waktu dalam log. Dari hasil pengukuran kinerja kemudian dilakukan identifikasi penggunaan *resource* melalui analisa terhadap grafik, untuk kemudian dilakukan *tuning* yang mencakup perubahan konfigurasi untuk kemudian dijalankan kembali.



Gambar 3.2 Langkah dan Strategi Eksperimen

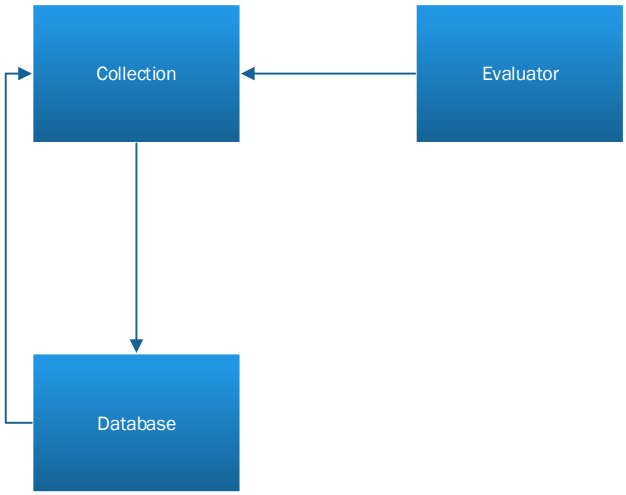
Adapun proses yang terjadi secara *high level* ditunjukkan oleh gambar di bawah ini



Gambar 3.3 Desain router yang digunakan pada eksperimen

Message simulator yang bertindak sebagai produser, akan mengirimkan sejumlah *message* (dalam satuan buah, misalnya 10000 buah *message*) yang dikirimkan kepada sebuah *queue*, selanjutnya untuk setiap *message* yang diambil dari *queue* oleh *message receiver*, akan dievaluasi kemanakah seharusnya *message* tersebut di-*route*, berdasarkan *routing rule* yang ada pada *database*. *Router* secara periodik mengambil *routing rule* dari *database* dan disimpan pada sebuah

collection sehingga *evaluator* tidak perlu mengambil *entry* ke *database* untuk setiap *message*, meminimalisir IO *operation*. Proses evaluasi ini adalah task per *message* yang dikerjakan oleh kerangka kerja *parallel processing* seperti Threadpool dan LMAX Disruptor.



Gambar 3.4 Proses evaluasi routing *message*

Routing rule pada *database* terdiri dari sebuah *table* dan terdiri dari *field* – *field* dengan penjelasan sebagai berikut:

Tabel 3.1 – Atribut di *database* pada *routing rules*

Nama	**Deskripsi**
Action_type	terdiri atas route, drop, dan forward (route ke lebih dari satu destinasi)
Activated	terrdiri atas 1 dan 0, menandakan apakah rule ini aktif atau tidak

Date_modified	menunjukkan date terakhir rule ini dimodifikasi
Description	deskripsi dari rule
Destination	nama dari destinasi queue
Destination_secondary	nama dari destinasi queue kedua
Destination_ternary	nama dari destinasi queue ketiga
Field_name	nama field yang menjadi key untuk matchmaking
Field_value	nama field yang menjadi value untuk matchmaking
Predicate	terdiri dari contains, endswith dan startswith

Untuk setiap *message* yang masuk, akan dilakukan *matchmaking* terhadap setiap *rule* dengan melakukan *loop*, sebagai contoh, sebuah *message* dengan struktur sebagai berikut,

Tabel 3.2 – Struktur *message* yang digunakan

Nama atribut	**Deskripsi**	**Contoh nilai**
UUID	Identifikasi unik antar *message*	*123e4567-e89b-12d3-a456-426655440000*
Sender	Pengirim *message*	Budi
Body	Isi dari *message* itu sendiri	Hello world!!!
addInfo	Informasi tambahan	+6280912121212
destination	Tujuan *message* saat proses routing	Foo-queue

Dari *message* yang masuk, akan dilakukan pencocokan terhadap *rule* yang memiliki atribut *field_name = body, predicate = contains dan field_value = fun*,

message tersebut akan cocok karena pada *body*, ia memiliki *string* yang mengandung (*contains*) *string fun*, sehingga ia akan diproses menurut *action type* dari *rule* tersebut. Sebagai contoh, apabila *action typenya* adalah *drop* maka *message* tidak akan di-*route*, apabila *action typenya route* maka *message* akan diset supaya destinasi nya sesuai dengan nilai dari *field* destinasi dari *rule* tersebut, nilai dari *field* destinasi *secondary* dan *ternary* akan diset apabila *action type* adalah *forward*.

1.3 Data Penelitian

Data pada penelitian ini berupa *message* yang terdiri dari nilai – nilai yang dipisahkan berdasarkan *delimiter* tertentu. Format *message* dalam kasus nyata tidak memiliki standar yang universal, karena tergantung dari arsitektur pengembang dan kebutuhan aplikasi yang menggunakannya. Untuk penelitian ini, penulis menggunakan *message* yang memiliki struktur seperti yang disebutkan pada tabel 3.2 di atas dengan nilai acak untuk masing – masing element

1.4 Variabel Penelitian

Berikut variabel yang digunakan dalam analisis penelitian ini:

1) Variabel tetap meliputi
    a. Kerangka kerja teknis yang digunakan (Threadpool dan LMAX Disruptor).
    b. Struktur *message* yang digunakan.

2) Variabel tak tetap meliputi

    a. Total *message* yang diterima

    b. Total dan persentase *success routed message*

    c. Total dan persentase *failed routed message*

    d. Persentase maksimum, rata – rata dan minimum penggunaan CPU

    e. Persentase maksimum, rata – rata dan minimum penggunaan memory

    f. Waktu yang dibutuhkan (dalam detik) untuk memroses seluruh *message*

3) Variabel *moderating* yang meliputi

    a. Jumlah *message* yang dikirimkan melalui *message simulator* dalam satuan buah (misalnya, 10 buah *messages*)

    b. Ukuran buffer untuk Threadpool (*working queue*) dan LMAX Disruptor (*RingBuffer*), dalam satuan *slot*, kedua kerangka kerja ini membungkus mekanisme level rendah dari *buffer* dan didesain supaya *programmer* dapat melakukan konfigurasi berdasarkan jumlah *slot* saja tanpa harus tahu berapa besar ukuran satu *slot* dalam satuan tertentu.

1.5 Teknik Analisis Data

Dalam penelitian ini, sejumlah *message* akan diberikan pada modul *router* melalui *queue* sebagai jalur komunikasi antar modul, selanjutnya *router* akan melakukan pencatatan variabel tak tetap ke dalam file untuk setiap detik. Data ini disimpan di dalam bentuk file agar memudahkan untuk diolah kemudian. Selanjutnya, data akan disajikan dalam bentuk tabular dan grafik dengan berfokus pada perbandingan:

1. Ukuran *buffer* dengan penggunaan CPU
2. Ukuran *buffer* dengan penggunaan *memory*
3. Ukuran *buffer* dengan *success rate*
4. Ukuran *buffer* dengan waktu pemrosesan seluruh *message*

Dengan teori yang telah disebutkan pada bagian sebelumnya, dugaan sementara adalah LMAX Disruptor memiliki waktu pemrosesan yang lebih cepat namun dengan kompensasi CPU *utilization* yang lebih tinggi daripada Threadpool. Seiring ukuran *buffer* yang ditingkatkan, Threadpool dapat menyamai waktu pemrosesan oleh Disruptor karena dalam sekali waktu, dapat menerima lebih banyak *message* sehingga mengurangi *rejected task*, sekaligus meningkatkan *success ratenya*. Kombinasi dari kedua *framework* ini memungkinkan terjadinya *switching* dari Threadpool ke LMAX Disruptor apabila sistem telah mencapai suatu *threshold* tertentu (contoh: *IF message_rate < n THEN use Threadpool ELSE use Disruptor*).