BAB I
PENDAHULUAN

1.1 Latar Belakang

Salah satu arsitektur perangkat lunak yang sedang populer saat ini adalah arsitektur *microservice*. *Microservice* itu sendiri adalah sebuah proses kohesif dan independen yang saling berinteraksi melalui pesan (Blinowski dkk., 2022). *Microservice* adalah pendekatan dalam pengembangan perangkat lunak yang memecah sebuah aplikasi menjadi bagian-bagian yang lebih kecil dan mandiri. Setiap bagian tersebut memiliki tugas dan fungsi yang spesifik, dan dapat berkomunikasi dengan bagian lainnya melalui antarmuka yang terdefinisi dengan jelas (Di Francesco dkk., 2019). Arsitektur *microservice* adalah aplikasi terdistribusi di mana semua modulnya adalah *microservice* (Blinowski dkk., 2022).

Arsitektur *microservice* memiliki beberapa kelebihan dibandingkan dengan arsitektur monolitik, di mana seluruh aplikasi dibangun dalam satu kesatuan. Pertama, arsitektur *microservice* memungkinkan pengembang untuk bekerja secara mandiri pada bagian-bagian tertentu dari aplikasi tanpa harus mempengaruhi bagian lainnya (Richards, 2022). Hal ini memungkinkan pengembangan dan perbaikan aplikasi menjadi lebih cepat dan efisien.

Selain itu, arsitektur *microservice* juga memungkinkan skalabilitas yang lebih baik. Dalam arsitektur monolitik, jika terdapat peningkatan jumlah pengguna atau permintaan, maka seluruh aplikasi harus ditingkatkan, bahkan jika hanya bagian tertentu yang membutuhkan peningkatan. Dalam arsitektur *microservice*, bagian-bagian yang membutuhkan peningkatan dapat ditingkatkan secara mandiri, sehingga aplikasi dapat lebih efisien dan responsif.

Salah satu kekurangan dari arsitektur *microservice* yang perlu dicermati adalah kompleksitas dalam manajemen infrastruktur dan koordinasi antar bagian aplikasi (Richards, 2022). Sebagai contoh, ketika sebuah aplikasi web menggunakan arsitektur *microservice*, maka setiap layanan harus berkomunikasi dengan layanan lainnya untuk memastikan semua bagian aplikasi dapat berjalan dengan lancar dan saling terintegrasi dengan baik. Ketika data yang ditransfer antar

layanan semakin besar, maka bisa terjadi *bottleneck* dalam komunikasi antar layanan tersebut. Hal ini dapat memperlambat respons waktu aplikasi, bahkan dapat menyebabkan kegagalan dalam operasional aplikasi.

Namun, kompleksitas dalam manajemen infrastruktur dan koordinasi antar bagian aplikasi dalam arsitektur *microservice* tidak selalu menghasilkan performa yang optimal (Blinowski dkk., 2022). Seperti yang telah dijelaskan sebelumnya, ketika data yang ditransfer antar layanan semakin besar, maka bisa terjadi *bottleneck* dalam komunikasi antar layanan tersebut. Salah satu solusi untuk mengatasi masalah ini adalah dengan mengoptimalkan proses komunikasi antar layanan dalam arsitektur *microservice*. Salah satu cara yang dapat dilakukan adalah dengan mengurangi ukuran data yang ditransfer antar layanan dengan mengompresinya. Pada umumnya format data yang sering digunakan dalam komunikasi antar layanan dalam arsitektur *microservice* adalah *JSON* (Velepucha & Flores, 2023). *JSON* merupakan format data yang ringan, mudah dibaca dan dipahami oleh manusia, dan juga mudah diinterpretasikan oleh mesin (Bourhis dkk., 2020). Namun, meskipun JSON tergolong ringan, ukuran data yang dikirimkan antar layanan dapat tetap besar ketika jumlah data yang ditransfer semakin banyak. Sebagai contoh pada json *array* di mana sebuah *array* memiliki beberapa json *object* yang semuanya memiliki *key* yang sama.

Metode kompresi yang dapat mengompresi objek JSON yang pertama adalah menggunakan metode HPack. HPack adalah protokol kompresi *header* HTTP yang dirancang khusus untuk digunakan dalam HTTP/2. HPack adalah sebuah metode kompresi yang menghilangkan *header redundant*, membatasi kerentanan terhadap serangan keamanan, dan memiliki persyaratan memori terbatas untuk digunakan dalam lingkungan yang terbatas (Krasic & Bishop, 2022).

HPack dapat di gunakan untuk mengatasi masalah komunikasi antar layanan, di mana jumlah *request* yang dilakukan oleh browser dalam satu waktu bisa sangat besar. Dalam situasi seperti ini, jika *response* dikirimkan dalam format yang tidak terkompresi, maka ukuran data yang dikirimkan dapat menjadi sangat besar. HPack pada penelitian ini bertujuan untuk mengurangi ukuran data *response*

dengan cara mengeliminasi redundansi dalam *json Object* dan mengurangi ukuran Index yang digunakan untuk mereferensikan kembali *response* yang sama.

Salah satu metode kompresi data yang populer adalah Gzip. Gzip merupakan metode kompresi data yang mengurangi ukuran data dengan cara menghilangkan redundansi dalam data. Dengan menggunakan Gzip, ukuran data yang dikirimkan antar layanan dapat dikurangi hingga ~95%(Objelean, 2011). Hal ini dapat mengurangi beban komunikasi antar layanan dalam arsitektur *microservice*, sehingga performa aplikasi dapat meningkat.

Dalam konteks implementasi HPack dan Gzip pada optimasi aplikasi dengan arsitektur *microservice*, penggunaan HPack dan Gzip dapat membantu mengoptimalkan proses komunikasi antar layanan dengan mengurangi ukuran data *response* yang dikirimkan, sehingga dapat mengurangi beban komunikasi antar layanan dalam arsitektur *microservice* dan meningkatkan performa aplikasi.

Dalam penelitian sebelumnya disimpulkan bahwa HPack yang saat di kombinasikan dengan Gzip untuk mengompresi *json Object* dapat mengurangi ukuran *json* sebanyak ~97% (Objelean, 2011).

Topik ini sangat menarik untuk diteliti karena masih sedikit peneliti yang membahas masalah ini. Selain itu, HPack juga belum banyak digunakan dan belum banyak dibahas tentang metode ini. Selain itu, belum ada peneliti yang mengombinasikan arsitektur *microservice* dan metode kompresi untuk tujuan optimasi. Dari penjelasan di atas, penulis akan membahas topik "Penerapan HPack dan Gzip dalam Optimalisasi Aplikasi dengan Arsitektur *microservice*" dalam penelitian ini.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka yang menjadi rumusan masalah dalam penelitian ini adalah :

1. Bagaimana implementasi dari HPack dan Gzip dalam mengompresi ukuran *response* pada arsitektur *microservice*?
2. Apakah kompresi *response* dari *microservice* dapat mengoptimalkan waktu respons dari aplikasi berbasis arsitektur *microservice*?

1.3 Batasan Masalah

Penelitian ini akan memfokuskan pada analisis dan evaluasi terhadap teknik kompresi data *JSON*, khususnya HPack dan Gzip, dalam rangka mengoptimalkan kinerja aplikasi dengan arsitektur *microservice*. Namun, ada beberapa batasan dalam penelitian ini yang perlu diperhatikan, yaitu:

1. Penelitian terbatas pada metode kompresi HPack dan Gzip; metode kompresi lainnya tidak akan dibahas secara detail.
2. Penelitian ini tidak akan membahas secara mendalam mengenai aspek keamanan yang terkait dengan penggunaan HPack dan Gzip.
3. Penelitian ini hanya membahas optimasi melalui kompresi *JSON* respons atau *requests*.
4. Penelitian ini hanya menggunakan durasi dan ukuran respons sebagai metrik yang di bahas.
5. Penelitian hanya akan mencakup aplikasi yang menggunakan arsitektur *microservice* dengan komunikasi *synchronous*; aplikasi dengan arsitektur lain, seperti monolitik, tidak akan dibahas di dalam penelitian ini.

1.4 Tujuan Penelitian

Berdasarkan latar belakang dan rumusan masalah yang telah dijelaskan sebelumnya, tujuan dari penelitian ini adalah

1. untuk mengoptimalkan proses komunikasi dalam lingkup ukuran dan kecepatan respons antar layanan dalam arsitektur *microservice* dengan menggunakan metode kompresi data.
2. Penelitian ini akan memfokuskan pada implementasi dan evaluasi performa khususnya pada kecepatan dan ukuran respons metode kompresi data Gzip dan HPack pada komunikasi antar layanan dalam aplikasi web berbasis arsitektur *microservice* yang menggunakan format data *JSON*.
3. Tujuan dari penelitian ini adalah untuk memperoleh hasil evaluasi ukuran dan durasi dari metode kompresi data Gzip dan HPack dalam meningkatkan kecepatan dan ukuran respons antar layanan dalam arsitektur *microservice*, sehingga dapat memberikan rekomendasi terbaik mengenai metode

kompresi data yang paling sesuai digunakan dalam lingkungan arsitektur *microservice* dengan format data *JSON*.

1.5 Manfaat Penelitian

Beberapa keuntungan yang dapat diperoleh dari penelitian ini adalah:

1. Memberikan pemahaman lebih mendalam tentang penggunaan arsitektur *microservice* dalam pengembangan aplikasi web, khususnya dalam hal manajemen infrastruktur dan koordinasi antar bagian aplikasi.
2. Memberikan solusi dalam mengurangi ukuran dan kecepatan respons dari komunikasi antar layanan dalam arsitektur *microservice* dengan mengurangi ukuran data yang ditransfer antar layanan dengan menggunakan metode kompresi data, seperti Gzip dan HPack.
3. Meningkatkan performa aplikasi web yang menggunakan arsitektur *microservice* dalam segi kecepatan dan ukuran respons dengan mengurangi beban komunikasi antar layanan melalui penggunaan metode kompresi data.
4. Memberikan rekomendasi kepada pengembang aplikasi web dalam pemilihan metode kompresi data yang tepat sesuai dengan kebutuhan dan karakteristik aplikasi yang dikembangkan.
5. Menjadi referensi dan acuan bagi penelitian selanjutnya yang terkait dengan penggunaan arsitektur *microservice* dan metode kompresi data dalam pengembangan aplikasi web.