BAB I
PENDAHULUAN

Bab pendahuluan ini bertujuan untuk memberikan gambaran singkat tentang pentingnya menjaga *server* agar tetap tersedia dan cepat. Pada revolusi industri 4.0 kebutuhan akan *server* menjadi sangat masif, mengingat semua sudah serba digital, cepat, dan otomatis. Oleh karena itu, pentingnya menjaga ketersediaan pada *server* dengan menggunakan rancangan infrastruktur *High Availability* dengan metode *Load Balancing* algoritma *Least Connection* serta menggunakan *alert system* berbasis *cloud* menjadi relevan dalam upaya ini.

1.1. Latar Belakang

Pada era Revolusi Industri 4.0 yang bertransformasi menuju *Society* 5.0, dunia memasuki masa yang serba cepat dan instan. Manusia sudah mulai hidup berdampingan dengan mesin canggih, saat ini teknologi telah digunakan di berbagai sektor untuk meningkatkan efisiensi dan produktivitas dengan melalui inovasi seperti *Robotics*, *Sensor Technology, Cloud Computing*, *Internet on Things*, dan *Artificial Intelligence* [1]. Pada awal tahun 2024 penggunaan internet mencapai 5,347 miliar pengguna, yang menunjukan peningkatan sebesar 28,91% dari 5 tahun sebelumnya [2]. Hal ini menunjukkan pentingnya infrastruktur teknologi, khususnya *Cloud Computing*, dalam mendukung komunikasi dan pertukaran data secara cepat dan efisien. Tanpa infrastruktur yang andal hal tersebut tidak akan berjalan dengan efektif [3].

Seiring dengan meningkatnya penggunaan internet, maka berbanding lurus dengan kebutuhan akan *server* dan *web server* yang juga ikut meningkat. Teknologi *Cloud Computing* pada saat ini memainkan peran penting dalam membantu individu maupun organisasi untuk mengelola dan memelihara *server* secara lebih efektif dan efisien tanpa perlu berinteraksi langsung dengan perangkat fisik. *Web server* digunakan untuk melayani permintaan (*request*) dengan protokol HTTP (*Hypertext Transfer Protocol*) dan mengirimkan tanggapan (*response*) berupa kode-kode dinamis ke *server* aplikasi. Namun, semakin banyak permintaan dari pengguna dapat mempengaruhi kinerja pada *server*. Jika permintaan meningkat

secara drastis, hal ini berpotensi menyebabkan lonjakan lalu lintas jaringan yang tinggi. Dalam kemungkinan terburuk, lonjakan tersebut dapat membuat *server* menjadi tidak dapat diakses atau bahkan mengalami mati (*down*), sehingga situs atau layanan yang bergantung pada *server* tersebut tidak dapat diakses [4].

Meskipun *cloud server* menjadi solusi bagi organisasi atau perusahaan yang membutuhkan pengelolaan *server* secara efisien, teknologi ini tidak sepenuhnya bebas dari kekurangan. Salah satu tantangan yang paling sering dihadapi adalah *Overload Traffic* dan *Request Cloud Environment*, yang terjadi ketika situs pada *server* diakses secara bersamaan oleh banyak pengguna dalam waktu yang singkat. Kondisi ini dapat mempengaruhi performa dan kinerja *server* secara keseluruhan, menyebabkan sistem menjadi lambat atau bahkan gagal (*fail*). Untuk mengantisipasi dan memulihkan sistem saat terjadi lonjakan lalu lintas dan permintaan pengguna yang tinggi, diperlukan pendekatan yang efektif dalam manajemen *server*, baik melalui optimasi infrastruktur maupun penerapan teknologi pendukung seperti *Load Balancing* dan *alert system* [5].

Untuk mengatasi tantangan seperti *Overload Traffic* dan memastikan layanan tetap tersedia di tengah tingginya permintaan, diperlukan *server* yang memiliki nilai ketersediaan yang tinggi, atau lebih dikenal dengan istilah *High Availability* (HA). Dalam konsep *High Availability*, teknologi seperti *Load Balancing* dapat diterapkan untuk mendistribusikan beban kerja secara seimbang di antara beberapa *server*, dengan syarat adanya lebih dari satu *server*. Teknologi ini dirancang untuk menyeimbangkan lalu lintas jaringan saat ada permintaan dari pengguna, sehingga dapat meningkatkan nilai ketersediaan pada sistem, mengurangi waktu respons, dan meringankan beban sistem agar mencapai performa maksimal.

Selain itu, jika salah satu *server* mengalami kegagalan, sistem tetap dapat berjalan dengan memanfaatkan *server* lain yang masih tersedia untuk menampung permintaan dari pengguna. *Load Balancing* juga didukung oleh berbagai metode dan algoritma, baik yang bersifat statis maupun dinamis, untuk memastikan beban pada sistem tetap seimbang secara optimal [6].

Pada penelitian sebelumnya, yang dilakukan oleh (Sumarna et al., 2019) membandingkan penerapan infrastruktur *n-clustering High Availability*

menggunakan *Load Balancing* dengan algoritma *Round Robin* dan *Least Connection* menggunakan HAProxy. Penelitian ini menggunakan metodologi *Cisco Lifecycle Services*, yang dikembangkan oleh Cisco untuk mendukung tahapan persiapan, perencanaan, perancangan, implementasi, pengujian, dan optimasi pada klaster. Dalam perancangan klaster, digunakan tiga lapisan (*layers*) dengan total 10 unit *server*, terdiri dari 4 *server* sebagai *Load Balancer* dan 6 *server* sebagai *backend server* untuk aplikasi. *Load Balance*r ditempatkan pada layer pertama dan kedua, sementara *backend server* berada pada layer ketiga. Pada konfigurasi HAProxy, setiap *server Load Balancer* dikonfigurasikan untuk menangani hingga maksimal 512 koneksi. Pengujian dilakukan dengan simulasi permintaan secara bersamaan (*load testing*) menggunakan aplikasi *Web Application Testing* (WAPT) untuk menganalisis performa algoritma *Round Robin* dan *Least Connection*. Hasil penelitian menunjukkan bahwa konfigurasi *double Load Balancing* dengan algoritma *Round Robin* memberikan hasil yang lebih baik, dengan rata-rata 2,81 sesi/detik, 3 sesi gagal, dan 3 halaman gagal. Sebagai perbandingan, algoritma *Round Robin* tunggal menghasilkan 2,82 sesi/detik, 4 sesi gagal, dan 4 halaman gagal, sementara algoritma *Least Connection* menghasilkan 2,84 sesi/detik, 8 sesi gagal, dan 8 halaman gagal [7].

Berdasarkan hasil penelitian sebelumnya, penelitian ini akan mengembangkan infrastruktur dengan pendekatan yang berbeda, yaitu menggunakan teknologi *Cloud Computing* dari penyedia layanan *cloud* (*cloud provider*) yang ditempatkan di berbagai zona geografis. Penempatan ini bertujuan untuk mengamati pengaruh *latency* dan *response time*, yang dapat bervariasi berdasarkan lokasi geografis *server*. Dalam penelitian ini, teknologi *Load Balancer* yang digunakan adalah Nginx dengan algoritma *Least Connection*, yang akan dibandingkan dengan ketahanan pada sistem *single* server. Berdasarkan penelitian lainnya, algoritma *Least Connection* dipilih karena lebih efektif dalam menangani permintaan (*request*) dengan tingkat pengguna yang tinggi secara bersamaan dengan mengalihkan lalu lintas jaringan pada *server* yang mempunyai koneksi aktif terendah. Sementara itu, Nginx dipilih sebagai alat *Load Balancer* karena penggunaan sumber daya yang lebih efisien dibandingkan dengan HAProxy, sehingga lebih sesuai untuk infrastruktur yang dirancang. Selain itu, untuk

mengantisipasi kegagalan sistem (*failures*), penelitian ini juga mengembangkan *backup server* dan *alert system* yang terintegrasi dengan Telegram melalui mekanisme Webhook. Sistem ini dirancang untuk memberikan pemberitahuan secara *real-time* kepada pengelola *server*, sehingga dapat segera mengambil tindakan yang diperlukan untuk memulihkan *server* dan menjaga ketersediaan layanan.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, dalam konteks implementasi *High Availability server* menggunakan *Load Balancer* dengan algoritma *Least Connection* dan *Alert System* berbasis *cloud,* maka dapat dibentuk beberapa rumusan masalah diantaranya:

1. Bagaimana rancangan infrastruktur *High Availability* menggunakan *Load Balancer* dan *Alert System* berbasis *Cloud Computing* untuk memastikan ketersediaan *server*?
2. Bagaimana pengaruh penggunaan *Load Balancer* dibandingkan dengan hanya menggunakan *single server* pada saat menghadapi lonjakan trafik?
3. Seberapa efektif penggunaan metode *Least Connection* pada *Load Balancer* dalam mendistribusikan beban kerja secara optimal?

1.3. Tujuan Penelitian

Penelitian ini bertujuan untuk:

1. Merancang infrastruktur *High Availability* (HA) multi zona berbasis *cloud* menggunakan *Load Balancer* untuk memastikan ketersediaan *server* yang andal.
2. Menganalisis pengaruh penggunaan *Load Balancer* dibandingkan dengan *single server* dalam menghadapi lonjakan trafik untuk meningkatkan kinerja *server*.
3. Menganalisis efektivitas metode *Least Connection* pada *Load Balancer* dalam mendistribusikan beban kerja secara optimal, termasuk peningkatan jumlah permintaan dan pengurangan *response time*.

1.4. Manfaat Penelitian

Penelitian ini diharapkan memberikan manfaat sebagai berikut:

1. Memberikan kontribusi pada pengembangan ilmu pengetahuan di bidang teknologi informasi, khususnya terkait infrastruktur *High Availability* (HA) berbasis *Cloud Computing*, *Load Balancing*, *Alert System* dan algoritma *Least Connection* dalam memastikan ketersediaan dan kinerja *server*.
2. Menjadi panduan bagi pengelola sistem IT dalam merancang dan mengimplementasikan infrastruktur *High Availability* (HA) yang andal menggunakan teknologi *Load Balancing* berbasis *cloud*.
3. Menggunakan solusi berbasis *Alert System* untuk mendeteksi dan menangani kegagalan *server* secara cepat dan efisien.

1.5. Batasan Masalah

Penelitian ini akan dibatasi dengan batasan masalah sebagai berikut:

1. Penelitian ini berfokus pada penerapan infrastruktur *High Availability* (HA) menggunakan Nginx sebagai *Load Balancer* dengan algoritma *Least Connection* dan Alert System yang terintegrasi dengan Telegram. Teknologi lain seperti HAProxy atau algoritma *Load Balancing* lain tidak dibahas secara rinci.
2. Implementasi dilakukan menggunakan layanan *cloud provider* tertentu seperti Google Cloud dan Alibaba Cloud dengan infrastruktur yang berada pada beberapa zona berbeda (*multi-zona*). Analisis mendalam terhadap aspek keamanan, infrastruktur *backend*, atau biaya *cloud* tidak termasuk dalam cakupan penelitian.
3. Pengujian kinerja *Load Balancer* dilakukan dengan metode *Load Testing* menggunakan perangkat lunak tertentu seperti K6 dan Grafana. Penelitian ini tidak mencakup implementasi pada *server* fisik atau *on-premises*.
4. *Server* yang digunakan pada penelitian ini hanya menggunakan spesifikasi tertentu, perbedaan spesifikasi dan skenario mungkin akan mempengaruhi kinerja dan hasil.

Halaman ini sengaja dikosongkan