

## BAB V

### KESIMPULAN DAN SARAN

#### 5.1 Kesimpulan

Penelitian ini bertujuan untuk mengoptimalkan proses load balancing pada cloud computing dengan menerapkan algoritma Ant Colony Optimization (ACO) sebagai pendekatan cerdas berbasis heuristik dan feedback. Sistem yang dibangun terdiri dari tiga komponen utama, yaitu load balancer (dengan implementasi ACO), dua backend server yang memproses request, dan client simulator menggunakan Locust untuk menghasilkan beban uji.

Hasil pengujian menunjukkan bahwa pendekatan ACO memiliki keunggulan adaptif yang mampu menyesuaikan distribusi permintaan secara real-time berdasarkan nilai pheromone dan kondisi aktual backend server (CPU usage). Sistem tidak hanya mendistribusikan request secara rata, tetapi juga secara cerdas memprioritaskan server yang memiliki performa lebih baik dan beban lebih ringan. Nilai pheromone yang terus diperbarui menjadi refleksi dari kondisi server, sehingga proses load balancing menjadi dinamis dan berbasis data historis.

Dari hasil uji performa, sistem mencatat rata-rata latensi sebesar 1919,90 ms, dengan latensi minimum 177,12 ms dan maksimum 6819,26 ms. Angka ini menunjukkan bahwa algoritma ACO efektif menjaga waktu respons tetap stabil pada sebagian besar kondisi, meskipun masih terdapat potensi lonjakan latensi pada skenario beban tinggi. Secara keseluruhan, implementasi ACO terbukti dapat meningkatkan efektivitas distribusi beban dibandingkan metode tradisional yang bersifat statis seperti round robin atau random.

## 5.2 Saran

Berdasarkan temuan dan analisis pada penelitian ini, terdapat beberapa saran yang dapat dipertimbangkan untuk pengembangan lebih lanjut:

### 1. **Tuning parameter ACO**

Melakukan pengujian dan penyesuaian lebih lanjut terhadap parameter algoritma, seperti tingkat evaporasi pheromone, nilai alpha dan beta, agar algoritma dapat lebih sensitif atau adaptif sesuai karakteristik beban nyata.

### 2. **Pengujian dengan skala lebih besar dan variasi skenario.**

Meningkatkan jumlah backend server serta variasi jenis permintaan (GET/POST dengan payload berbeda) agar sistem lebih siap diterapkan di lingkungan produksi yang kompleks dan dinamis.

### 3. **Integrasi monitoring dan dashboard real-time.**

Menambahkan fitur visualisasi kondisi backend server, nilai pheromone, dan statistik performa, sehingga administrator dapat memantau dan menganalisis performa sistem secara langsung.