ABSTRACT

A server is a device that provides services, both in the form of hardware and software. The existence of a server allows for the provision of resources such as data. Another function of the server is to be a place to deploy web applications. A server should be set to have availability so that when a client needs data or wants to access a web application, the service will still be available. One way to maintain server availability is to implement a load balancer. A load balancer is a service that can distribute traffic to several servers so that certain servers will not be overloaded. Load balancers whose services can be used include NGINX, with round-robin, least-connection, and ip-hash methods. Each method has a different way of working and its Quality of Service can be analyzed to determine the appropriate method. In addition, server availability can be monitored in real-time using Grafana and Prometheus to ensure that the server is working properly or there are obstacles that need to be overcome. For this reason, a load balancer was implemented on the flask web app to improve web server performance, as well as integrating with Grafana and Prometheus services so that they can be monitored. From the research results, it is known that the load balancer can divide traffic into 3 VMs using the round robin, least connection, and IP hash methods, and monitored on the Grafana dashboard with the Prometheus data source. Of the three methods, the largest throughput uses the IP hash method (0.571 Mbps Flask C) and round robin (0.215 Mbps Flask A, 0.281 Flask B, 0.241 Flask C). Then the smallest packet loss uses the least connection method (0.5% Flask A, 0.3% Flask B, 0.4% Flask C). Then the smallest delay uses the round robin method (7.362 ms Flask A, 5.549 ms Flask B, 7.992 Flask C). Finally, the smallest jitter uses the round robin method (4.092 ms Flask A, 1.735 ms Flask B, 3.822 Flask C).

***Keywords**: load balancer, Flask, nginx, grafana, prometheus*