Integrasi Large Language Model pada Sistem Monitoring Menggunakan Model Context Protocol untuk
Analisis Gangguan Layanan Microservices
Muhammad Silmy Firdaus, Dr. Sahirul Alam, S.T., M.Eng.
Universitas Gadjah Mada, 2026 | Diunduh dari http://etd.repository.ugm.ac.id/

INTISARI

INTEGRASI *LARGE LANGUAGE MODEL* PADA SISTEM *MONITORING* MENGGUNAKAN *MODEL CONTEXT PROTOCOL* UNTUK ANALISIS GANGGUAN LAYANAN *MICROSERVICES*

Muhammad Silmy Firdaus

22/506005/SV/22032

Penerapan arsitektur *microservices* pada lingkungan *container* Docker menawarkan skalabilitas tinggi, tetapi menghasilkan volume data telemetri yang besar dan tersegmentasi sehingga metode pemantauan konvensional tidak mampu mendukung analisis akar penyebab gangguan secara cepat. Proyek akhir ini merancang dan mengimplementasikan sistem *Monitoring* yang mengintegrasikan *Large Language Model* (LLM) melalui *Model Context Protocol* (MCP) untuk mengotomatisasi analisis gangguan pada layanan *microservices* berbasis Docker. MCP *Server* kustom dikembangkan untuk menjembatani dua sumber telemetri, yaitu metrik *container* yang dikumpulkan Prometheus melalui cAdvisor serta log *container* yang diambil dari Docker Daemon melalui Docker SDK, dengan antarmuka bahasa alami yang disediakan oleh LLM melalui Cursor sebagai MCP *Client*. Validasi akurasi telemetri terhadap delapan *container* Docker *e-commerce* menghasilkan rerata MAE 0,1071 MB dan rerata MAPE 0,29% dengan seluruh *container* berada di bawah ambang batas 1%, sehingga keandalan data terkonfirmasi. Selanjutnya, integrasi NLP melalui LLM terbukti layak secara teknis untuk mendukung analisis gangguan. Pengujian waktu respons menunjukkan sistem merespons dalam rentang 3-23 detik dengan rerata 5 detik (ringan), 9,2 detik (sedang), dan 12,8 detik (berat). Komponen *reasoning* LLM mendominasi 98,72%-99,81% total waktu respons, sedangkan MCP *Server* hanya berkontribusi *overhead* 0,044 detik. Pengujian skalabilitas dengan 1-10 *Concurrent User* menunjukkan penggunaan CPU yang elastis dengan rata-rata puncak 23,01% pada 10 *user* serta RAM stabil pada kisaran 165,7-169,3 MB berkat arsitektur *stateless* MCP *Server*. Hasil penelitian ini mengonfirmasi bahwa integrasi LLM berbasis MCP layak secara teknis untuk mengotomatisasi analisis insiden.

Kata kunci: *Model Context Protocol*, *large language model*, *microservices*, Docker, *root cause analysis*.

UNIVERSITAS
GADJAH MADA
**Integrasi Large Language Model pada Sistem Monitoring Menggunakan Model Context Protocol untuk
Analisis Gangguan Layanan Microservices**
Muhammad Silmy Firdaus, Dr. Sahirul Alam, S.T., M.Eng.
Universitas Gadjah Mada, 2026 | Diunduh dari http://etd.repository.ugm.ac.id/

ABSTRACT

INTEGRATION OF LARGE LANGUAGE MODELS INTO MONITORING SYSTEMS USING THE MODEL CONTEXT PROTOCOL FOR MICROSERVICES DISRUPTION ANALYSIS

Muhammad Silmy Firdaus

22/506005/SV/22032

The adoption of microservices architecture in Docker-based container environments offers high scalability yet generates vast volumes of segmented telemetry data, rendering conventional Monitoring inadequate for rapid root cause analysis. This project designed and implemented a system that integrates a Large Language Model (LLM) through the Model Context Protocol (MCP) to enable automated disruption analysis on Docker-based microservices. A custom MCP Server was developed to bridge two telemetry sources container metrics from Prometheus via cAdvisor and container logs from the Docker Daemon via Docker SDK with a natural language interface provided by the LLM through Cursor as the MCP Client. Telemetry accuracy validation on eight e-commerce Docker containers yielded a mean MAE of 0.1071 MB and mean MAPE of 0.29%, with all containers below the 1% threshold, confirming data reliability. NLP integration through the LLM also proved technically feasible for disruption analysis. Response time testing showed the system responded within 3-23 seconds, averaging 5 seconds for light, 9.2 seconds for moderate, and 12.8 seconds for heavy queries. The LLM reasoning component dominated 98.72%-99.81% of total response time, while the MCP Server contributed only 0.044 seconds of overhead. Scalability testing with 1-10 Concurrent Users revealed elastic CPU utilization averaging 23.01% peak at 10 users and stable RAM consumption of 165.7-169.3 MB due to the stateless MCP Server architecture. These findings confirm that MCP-based LLM integration is technically viable for automating incident analysis.

Keywords: *Model Context Protocol, large language model, microservices,* Docker*, root cause analysis*