Indian Institute of Technology Jodhpur

Fundamentals of Distributed Systems

Assignment – 1

Dynamic Load Balancing for a Smart Grid

Submitted By:
Anusha V
G24AI2042

1. Project Title

Dynamic Load Balancing for a Smart Grid

2. Objective

To design and build a scalable system for a Smart Grid that dynamically balances Electric Vehicle (EV) charging requests across multiple substations based on their real-time load, complete with a comprehensive observability stack.

3. Technologies Used

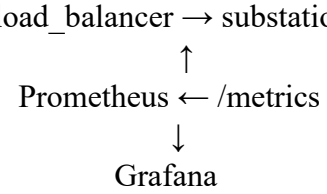
- **Language**: Python 3
- **Microservices**: Flask
- **Monitoring**: Prometheus, Grafana
- **Containerization**: Docker, Docker Compose

4. Architecture

The system contains:

- charge_request_service: public-facing service to accept EV charging requests
- load_balancer: polls substation load and routes requests to the lightest one
- substation_service (3 replicas): simulate load and expose metrics
- prometheus: scrapes substation metrics every 5s
- grafana: visualizes substation loads in a live dashboard

Diagram:



5. Implementation Highlights

- Each substation handles charging and updates its load
- Exposes real-time load at /metrics using prometheus_client
- The load balancer polls metrics and forwards each EV to the least loaded substation
- Prometheus scrapes all substations and feeds Grafana
- Grafana provides live visibility with substation_current_load

6. Log Output and Explanation

Sample Terminal Output (from test.py)

```
' 'c:\Users\Anusha\.vscode\extensions\ms-python.debugpy-2025.8.0-win32-x64\bundled\libs\debugpy\launcher' '51262' '--' 'c:\Users\Anusha\Documen
ts\IITJ_T2\FDS\smart---grid---load---balancer\load_tester\test.py'
EV 0 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 1 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 2 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 3 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 4 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 5 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 6 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 7 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 8 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 9 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 10 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 11 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 12 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 13 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 14 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 15 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 16 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 17 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 18 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 19 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 20 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 21 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 22 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 23 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 24 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 25 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 26 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 27 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 28 → 200: Routed to: http://substation1:8000/charge | Charging started
EV 29 → 200: Routed to: http://substation1:8000/charge | Charging started
PS C:\Users\Anusha\Documents\IITJ_T2\FDS\smart---grid---load---balancer>
```

Load Distribution:

Each substation shows POST /charge in logs, proving successful balancing.

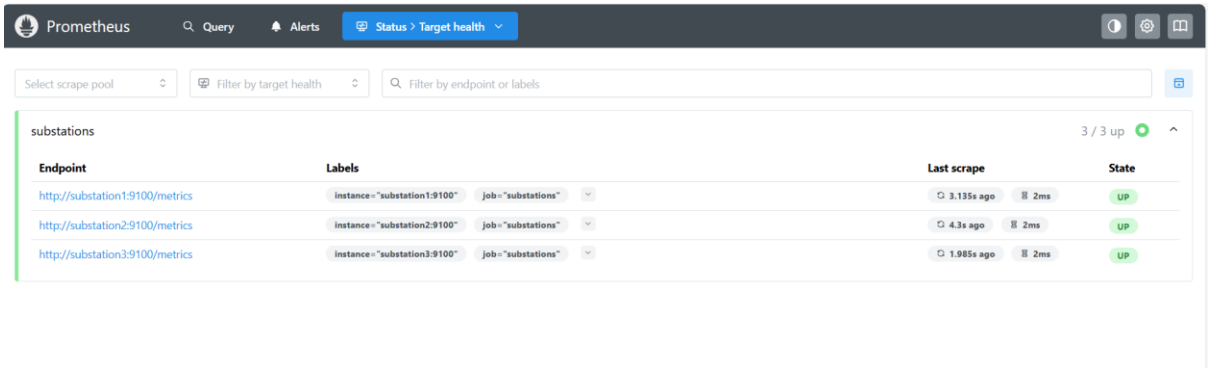
Figure: Docker Compose terminal showing successful charge request routing to substation1, along with Prometheus scraping substation2 and substation3. This confirms distributed request handling.

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

charge_request  | 172.21.0.1 - - [22/Jun/2025 12:55:53] "POST /charge HTTP/1.1" 200 -
load_balancer   | 172.21.0.8 - - [22/Jun/2025 12:55:54] "POST /request_charge HTTP/1.1" 200 -
substation1     | 172.21.0.7 - - [22/Jun/2025 12:55:54] "POST /charge HTTP/1.1" 200 -
charge_request  | 172.21.0.1 - - [22/Jun/2025 12:55:54] "POST /charge HTTP/1.1" 200 -
substation1     | 172.21.0.7 - - [22/Jun/2025 12:55:54] "GET /metrics HTTP/1.1" 200 -
substation2     | 172.21.0.7 - - [22/Jun/2025 12:55:54] "GET /metrics HTTP/1.1" 200 -
substation3     | 172.21.0.7 - - [22/Jun/2025 12:55:54] "GET /metrics HTTP/1.1" 200 -
substation1     | 172.21.0.7 - - [22/Jun/2025 12:55:55] "POST /charge HTTP/1.1" 200 -
load_balancer   | 172.21.0.8 - - [22/Jun/2025 12:55:55] "POST /request_charge HTTP/1.1" 200 -
charge_request  | 172.21.0.1 - - [22/Jun/2025 12:55:55] "POST /charge HTTP/1.1" 200 -
```

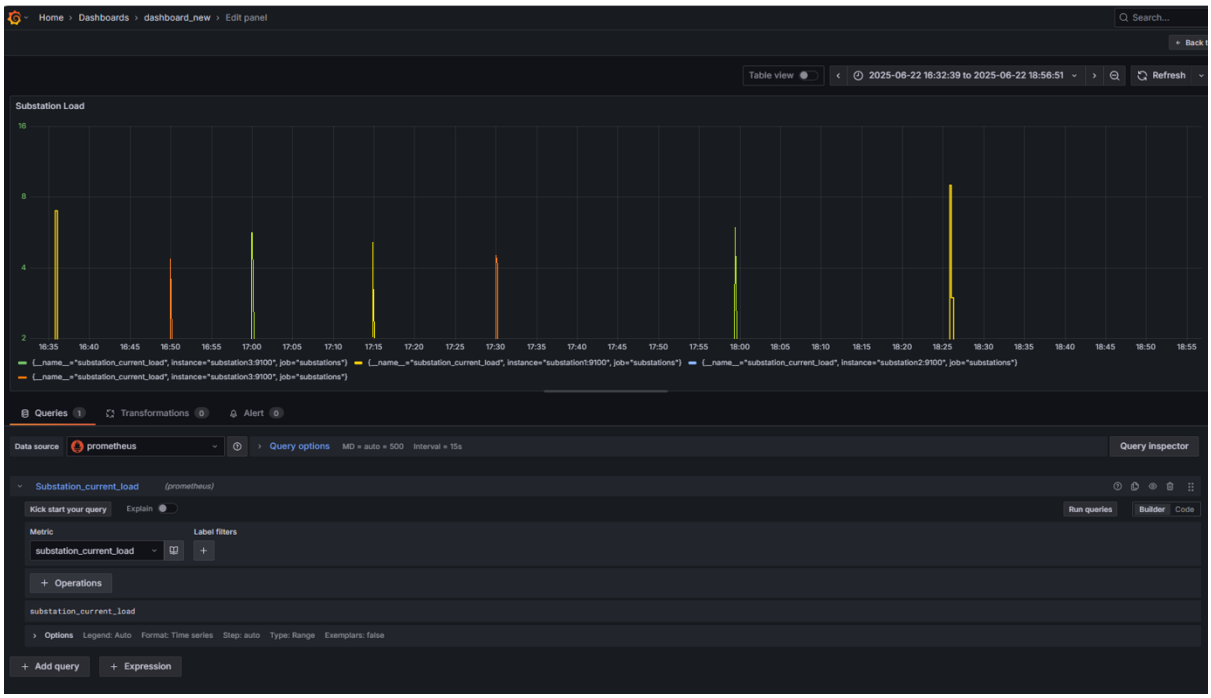
Prometheus Logs:

Successfully scrapes each substation



7. Grafana Dashboard

- The Grafana panel below visualizes the real-time values returned by the Prometheus metric substation_current_load.
- Each line represents one substation instance (e.g., substation1:9100, substation2:9100, etc.) and tracks its load over time. This confirms that Prometheus is not only scraping the values successfully, but the values also reflect dynamic updates from EV charging sessions.
- Graph Observations:
  - Y-axis shows the number of active charging sessions
  - X-axis shows the timestamp
- A visible spike proves that EVs were actively routed and processed



8. Load Testing

Using the test.py script in load_tester/, we simulated 30 electric vehicle (EV) charging requests with random intervals. The load balancer successfully distributed these requests to the least loaded substations in real time.

Refer Image of section 7

8. Demo Video Link

https://youtu.be/AYG0GsMnDk0

9. Conclusion

This project successfully demonstrates a scalable, containerized Smart Grid Load Balancer using Python microservices, Prometheus for real-time monitoring, and Grafana for intuitive visualization. The system efficiently routes EV charging requests to the least loaded substation, ensuring optimal resource usage and avoiding overload conditions.

Dynamic load balancing was verified through terminal logs, Prometheus queries, and Grafana dashboards. All components — including three substation services, a central load balancer, and public-facing API — were containerized and coordinated using Docker Compose.

The system is modular, fault-tolerant, and observability-driven, making it well-suited for real-world applications in energy distribution, smart infrastructure, and dynamic resource allocation systems.