Project for the Computer Networks Course

Course Instructor: Dr. Sadeghian

Prepared by: Amir Behnam

Project Description: CPU Monitoring System with Prometheus

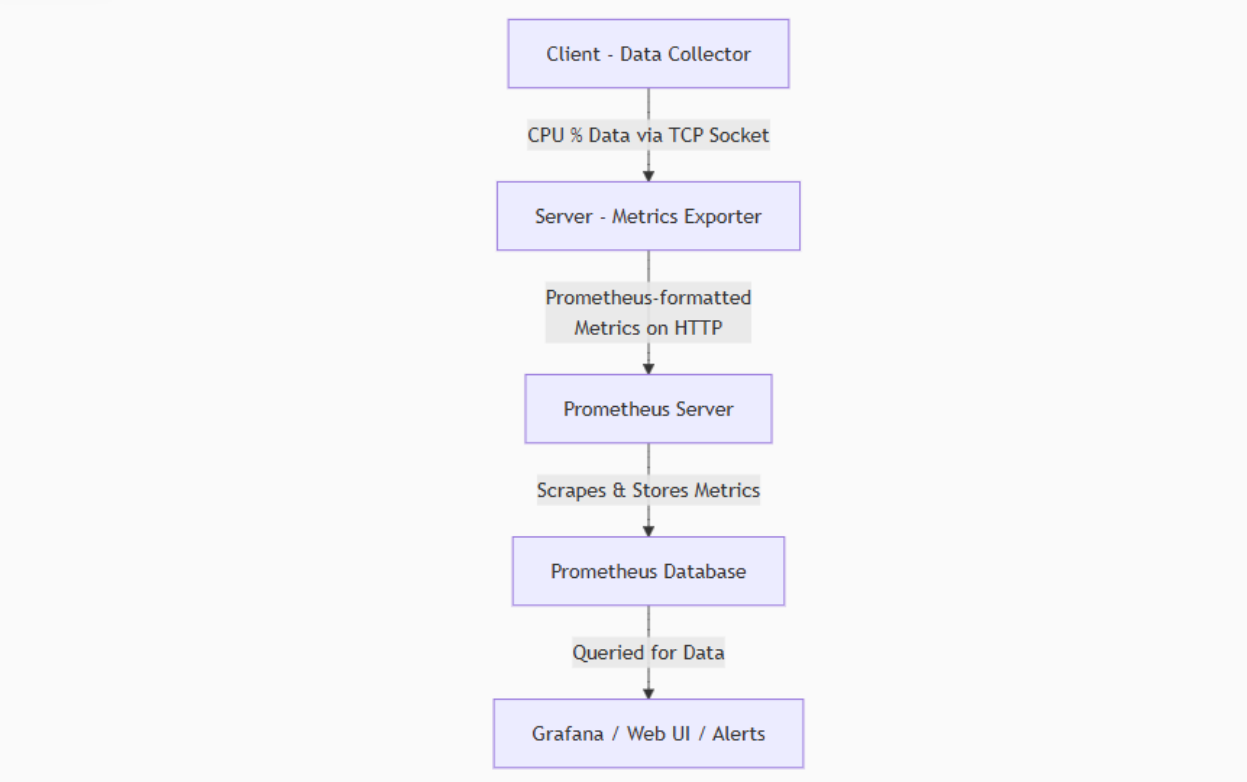
This project is a client-server application designed to monitor CPU usage in real-time. It collects CPU performance data from a machine and exposes it as metrics for **Prometheus**, a powerful monitoring system and time-series database. This allows you to visualize the CPU usage over time, set up alerts, and create dashboards (e.g., using Grafana).

How It Works: A Step-by-Step Breakdown

The system consists of three main components:

1. **The Client (**client.py**)** - The Data Collector
2. **The Server (**server.py**)** - The Metrics Exporter
3. **Prometheus (**prometheus.yml**)** - The Monitoring Backend

Here is the flow of data between these components:



Code

1. The Client (Data Collector)

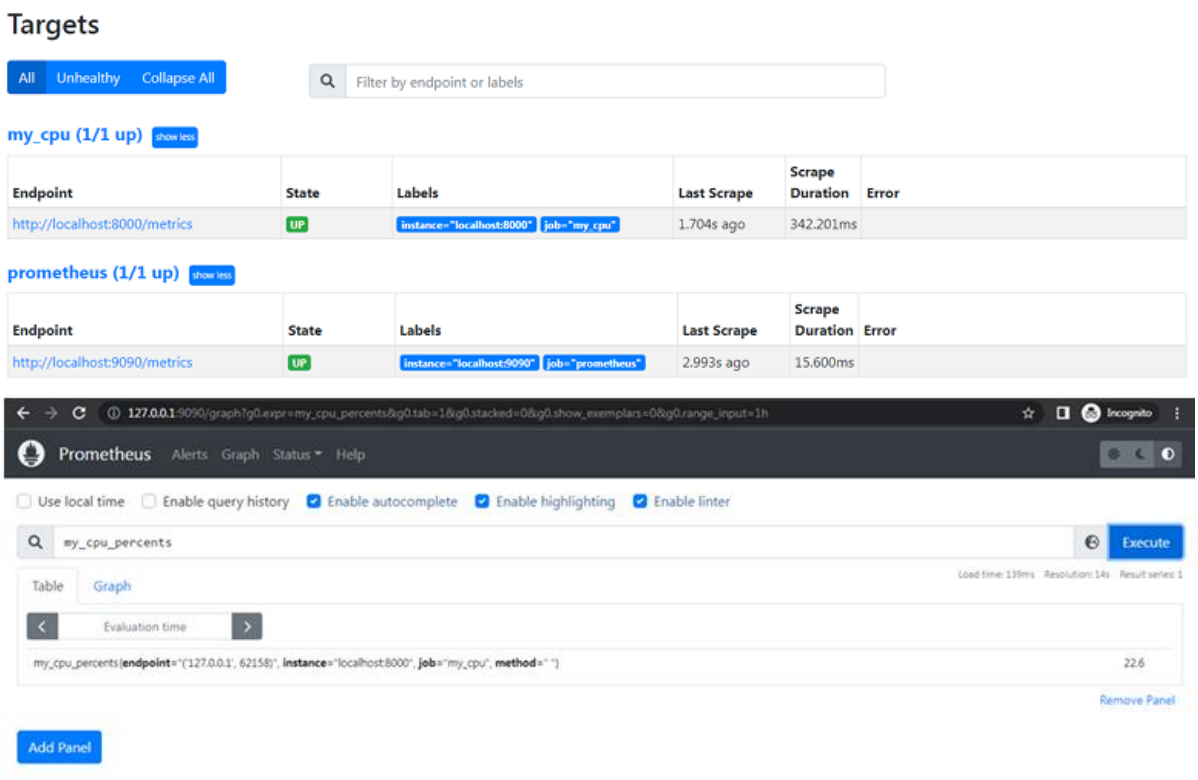
- **Purpose:** Runs on the machine you want to monitor. Its job is to continuously measure CPU usage and send that data to the server.
- **Operation:**
    1. It initiates a persistent **TCP socket connection** to the server's IP address (127.0.0.1) and port (8080).
    2. In an infinite loop, it uses the psutil library to get the current **CPU utilization percentage** (averaged over 1 second).
    3. It packages this percentage into a JSON object (e.g., {"The Perecentage Of CPU USAGE" : 12.5}).
    4. It sends this JSON data over the socket to the server and waits for a response (which it prints out).

2. The Server (Metrics Exporter & Bridge)

- **Purpose:** Acts as a bridge between the client and Prometheus. It receives raw CPU data and converts it into a format Prometheus can understand.
- **Operation:**
    1. It starts a **TCP socket server** on port 8080 to listen for incoming connections from clients.
    2. **Concurrently**, it starts an **HTTP server** on port 8000 using prometheus_client. This server provides a /metrics endpoint which Prometheus will scrape.
    3. For each client that connects, the server starts a new thread to handle the communication without blocking others.
    4. It receives the JSON data from the client, decodes it, and extracts the CPU percentage value.
    5. It updates a **Prometheus Gauge** metric named my_cpu_percents with this new value. A Gauge is a metric that represents a single numerical value that can go up and down, perfect for CPU usage.
    6. The metric is labeled with the client's socket information (its IP and port), allowing you to monitor multiple clients if you extend the project.
- Program output in Server & Client:

3. Prometheus (Monitoring Backend)

- **Purpose:** To scrape, store, and query the metrics provided by the server.
- **Configuration (**prometheus.yml**):**
    - The configuration file defines a **scrape job** named "my_cpu".
    - This job tells Prometheus to scrape the metrics from the target localhost:8000 (the HTTP server started by server.py).
    - Prometheus will automatically poll the /metrics endpoint on this port every 15 seconds (as per the scrape_interval), collect the current value of the my_cpu_percents gauge, and store it in its time-series database.
- **Program output in Prometheus:**



Key Design Choice: Why a Gauge?

As detailed in your report, the choice of a **Gauge** metric type was intentional and correct:

- **Counter vs. Gauge:** A Counter is designed to only increase (e.g., total number of requests, errors, or tasks completed). It is used with the rate() function to see how fast it is increasing. A Gauge can arbitrarily go up and down, which is the exact nature of CPU utilization percentage (e.g., 14.6%, 7.3%, 35.7%).
- **Summary/Histogram vs. Gauge:** Summary and Histogram metrics are used to track the *distribution* of observations, typically request latencies. They are optimized for calculating quantiles (e.g., 95th percentile). They are not suitable for reporting a single, current state value like CPU usage. The Gauge is the appropriate tool for this job.

Summary

In essence, this project creates a pipeline:
CPU Usage → Client → Socket → Server → Prometheus Gauge → HTTP Endpoint → Prometheus Server.

Once the data is in Prometheus, it becomes available for building dashboards, analyzing trends, and setting up alerts to notify you if the CPU usage exceeds a certain threshold for a period of time.