Custom System Metrics Exporter

R.Rohith
EP21B030

The application collects system metrics from `/proc/meminfo` and the `iostat` command, exposing them on port 18000 for Prometheus to scrape.

- Before running the application, make sure you have Flask installed.
- The application uses **subprocess** to run the `iostat -k` command and saves the standard output to the file `metrics/iostat.txt` every second.
- It opens the `/proc/meminfo` file every second and writes it to the file `metrics/meminfo.txt`.
- It parses both using **RegEx** and uses **Flask** to make the metrics available at:
  - `http://localhost:18000/metrics` (for Prometheus scraping)
  - `http://localhost:18000/health` (for health checks)

Make sure to add the following to your Prometheus YAML file to configure it to scrape from `http://localhost:18000/metrics` every 2 seconds.

```yaml
- job_name: "custom_metrics"
  scrape_interval: 2s
  static_configs:
    - targets : ["localhost:18000"]
```

1 Available Metrics

1.1 CPU Metrics

- `cpu_avg_percent{mode="user"}` - CPU time spent in user mode
- `cpu_avg_percent{mode="nice"}` - CPU time spent in user mode with low priority
- `cpu_avg_percent{mode="system"}` - CPU time spent in system mode
- `cpu_avg_percent{mode="iowait"}` - CPU time spent waiting for I/O operations
- `cpu_avg_percent{mode="idle"}` - CPU idle time

1.2 I/O Metrics

For each device:

- `io_read_rate{device="<device>"}` - Read operations per second (in bytes/sec)
- `io_write_rate{device="<device>"}` - Write operations per second (in bytes/sec)
- `io_tps{device="<device>"}` - Transfers per second
- `io_read_bytes{device="<device>"}` - Total bytes read
- `io_write_bytes{device="<device>"}` - Total bytes written

1.3 Memory Metrics

All metrics from `/proc/meminfo` are exposed with the prefix `meminfo_`, for example:

- `meminfo_memtotal` - Total usable RAM
- `meminfo_memfree` - Free memory
- `meminfo_buffers` - Memory used by kernel buffers
- `meminfo_cached` - Memory used for page cache
- Many other memory-related metrics