Telegraf Prometheus Metrics - Units of Measurement Analysis

Based on your Telegraf configuration using the **inputs.cgroup**, inputs.net, and **inputs.processes** plugins, I've analyzed each metric from your Prometheus data sample to identify their units of measurement. Here's the complete breakdown:

CPU Statistics (cgroup plugin)

The CPU metrics use **microseconds (µs)** as their unit of measurement:[1] [2] [3]

- `container_cgroup_cpu_stat_system_usec`: 27,356,391 microseconds - represents CPU time spent in kernel/system mode
- `container_cgroup_cpu_stat_user_usec`: 31,647,024 microseconds - represents CPU time spent in user mode

These values correspond to the cgroup v2 `cpu.stat` file entries for `system_usec` and `user_usec`.[2] [3]

I/O Statistics (cgroup plugin)

All I/O metrics are measured in **bytes**:[4] [5]

- `container_cgroup_io_stat_*_rbytes`: Read bytes from specific devices (252:0, 252:1, 8:16)
- `container_cgroup_io_stat_*_wbytes`: Write bytes to specific devices

These represent cumulative byte counters for I/O operations per block device.[5] [4]

Memory Statistics (cgroup plugin)

Memory metrics are all measured in **bytes**:[6] [7] [8]

- `container_cgroup_memory_current`: 55,484,416 bytes (current memory usage)
- `container_cgroup_memory_max`: 1,073,741,824 bytes (1GB memory limit)
- `container_cgroup_memory_swap_current`: 8,249,344 bytes (current swap usage)
- `container_cgroup_memory_swap_max`: 1,073,741,824 bytes (1GB swap limit)

Network Statistics (net plugin)

Network metrics use **bytes** for data transfer and **count** for packets/errors:[9] [10] [11]

Bytes (cumulative counters):

- `container_net_bytes_recv`: 67,415,307 bytes received
- `container_net_bytes_sent`: 45,881,235,112 bytes sent

Counts (cumulative counters):

- `container_net_packets_recv`: 1,020,937 packets received
- `container_net_packets_sent`: 1,047,902 packets sent
- `container_net_drop_in/out`: Dropped packet counts
- `container_net_err_in/out`: Error packet counts

Important Note: Network metrics are **cumulative** since container start, not rates per second.[11] [12]

Process Statistics (processes plugin)

Process metrics are measured as **instantaneous counts**:[13] [14]

- `container_processes_running`: 0 - currently running processes
- `container_processes_sleeping`: 8 - sleeping processes
- `container_processes_total`: 8 - total processes
- `container_processes_zombies`: 0 - zombie processes

These represent the current state snapshot of processes at collection time.[13]

Key Insights for Monitoring

1. **CPU times** are cumulative microseconds since container start
2. **Memory values** are current usage/limits in bytes
3. **I/O statistics** are cumulative byte counters per device
4. **Network metrics** are cumulative since container start - use rate functions for bandwidth calculations[12] [11]
5. **Process counts** are instantaneous snapshots

For rate calculations (like bandwidth), you'll need to apply derivative or rate functions in your monitoring system (Prometheus `rate()` function or similar).[11] [12]

⁂

1. https://www.influxdata.com/integration/cgroup/
2. https://stackoverflow.com/questions/54636124/sending-metrics-from-telegraf-to-prometheus
3. https://github.com/prometheus-community/systemd_exporter/issues/34
4. https://pkg.go.dev/github.com/shanas-swi/telegraf-v1.16.3/plugins/inputs/cgroup

5. https://grafana.com/docs/grafana-cloud/send-data/metrics/metrics-influxdb/push-from-telegraf/
6. https://facebookmicrosites.github.io/cgroup2/docs/cpu-controller.html
7. https://docs.influxdata.com/telegraf/v1/plugins/
8. https://pkg.go.dev/github.com/influxdata/telegraf/plugins/inputs/prometheus
9. https://www.kernel.org/doc/Documentation/cgroup-v2.txt
10. https://archive.docs.influxdata.com/telegraf/v1.7/plugins/inputs/
11. https://stackoverflow.com/questions/65551215/get-docker-cpu-memory-limit-inside-container
12. https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/6/html/resource_management_guide/sec-memory
13. https://andrestc.com/post/cgroups-io/
14. https://stackoverflow.com/questions/74796436/rss-memory-equivalent-in-cgroup-v2
15. https://www.kernel.org/doc/Documentation/cgroup-v1/memory.txt
16. https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/6/html/resource_management_guide/ch-subsystems_and_tunable_parameters
17. https://stackoverflow.com/questions/50865763/memory-usage-discrepancy-cgroup-memory-usage-in-bytes-vs-rss-inside-docker-con
18. https://facebookmicrosites.github.io/cgroup2/docs/io-controller.html
19. https://www.usenix.org/system/files/lisa21_slides_down.pdf
20. https://docs.oracle.com/en/operating-systems/oracle-linux/6/adminsg/ol_memory_cgroups.html
21. https://linuxera.org/cpu-memory-management-kubernetes-cgroupsv2/
22. https://cloud.megaease.cn/docs/manual/90.metrics-collection/01.host
23. https://stackoverflow.com/questions/46704745/docker-stat-network-traffic/46706595
24. https://github.com/Mirantis/telegraf/blob/master/plugins/inputs/system/PROCESSES_README.md
25. https://github.com/influxdata/telegraf/issues/6187
26. https://gist.github.com/aiwantaozi/45511575d47101b49a8883acf19ab394
27. https://stackoverflow.com/questions/48702845/count-the-number-of-running-process-with-telegraf
28. https://community.grafana.com/t/value-are-cumulativ-problem-how-to-solve/70829
29. https://docs.cloudera.com/management-console/1.5.4/monitoring-metrics/topics/cdppvc_ds_container_network_transmit_packets_total_visor.html
30. https://www.influxdata.com/blog/how-to-time-data-collection-telegraf/
31. https://forum.opnsense.org/index.php?topic=29950.0
32. https://community.grafana.com/t/vsphere-datastores-usage-capacity-shows-no-data/18755
33. https://www.influxdata.com/blog/collecting-running-process-counts-with-telegraf/
34. https://ppl-ai-code-interpreter-files.s3.amazonaws.com/web/direct-files/0342f68306f19478e870af2d7c296f8d/1bf17ff5-2871-4839-b881-f59f03029363/e2c65dd1.csv