Performance Testing Report

Objective

The purpose of load testing and performance validation is to evaluate the system's ability to handle high log ingestion rates while maintaining stability and efficiency. This test aims to:

- Validate capacity limits by determining the maximum events per second (EPS) the system can process before performance starts to degrade.
- Ensure system stability under normal and peak workloads.
- Identify bottlenecks in log processing, query response time, and resource usage.
- Verify Graylog's alerting and indexing performance under stress conditions.

KPIs & Baselines

The following Key Performance Indicators (KPIs) will be used to measure system performance:

Log Ingestion Rate (EPS - Events Per Second)

System should be able to sustain the expected EPS without performance degradation.

- Production Target: 10,000 EPS
- Prototype Target: 500–1,000 EPS

Query Response Time

We'll run queries at various load levels to check response time.

- Target: Queries should return results within 5 seconds.

System Resource Usage

System metrics will be monitored during testing.

- Target:
    - CPU Usage: Graylog and Elasticsearch should not exceed 80% CPU usage.
    - Memory Usage: Should remain below 70%.
    - Disk I/O Utilisation: Should not exceed 85% to avoid slowdowns.

Log Processing Latency

We'll inject logs and measure the delay before they appear in search results.

- Target: Logs should be indexed and searchable within 3-5 seconds of ingestion.

Alert Generation Speed

We'll simulate security events and measure the delay before alerts appear.

- Target: Security alerts should trigger within 5 seconds of event detection.

System Stability & Failure Rate

We'll gradually increase EPS beyond expected loads until failures appear.

- Target: No crashes, missed logs, or significant slowdowns.

Test Scenarios

To evaluate the system's performance, logs will be generated using Logstash and injected into Graylog. Different test batches will be used to simulate varying log loads and analyse the system's response.

1. Baseline Performance Test

- **Objective:** Establish normal system behavior under expected production loads.
- **Method:**
  - Use Logstash to generate 500 EPS for 10 minutes.
  - Measure CPU, memory, disk usage, and query response time.
  - Check if logs are indexed within 5 seconds.
  - Monitor alert generation speed.

2. Stress Test - Increasing Log Volume

- **Objective:** Determine the maximum log ingestion capacity before performance degradation.
- **Method:**
  - Start at 600 EPS and gradually increase by 100 EPS every 3 minutes.
  - Continue until system bottlenecks appear (e.g., delays, dropped logs, resource exhaustion).

  - Record the highest sustainable EPS before failures occur.

3. Query Performance Under Load

- **Objective:** Test query response time under high system load.
- **Method:**
  - Inject 1,000 EPS for 10 minutes.
  - Simultaneously execute complex search queries in Graylog.
  - Measure query response time and compare against baseline.

Results & Findings

Test Scenario 1: Baseline Performance Test

Steps Taken:

- I configured Logstash to generate 500 Events Per Second (EPS) continuously for a duration of 10 minutes to simulate a minimal operational workload.
- During the test, I monitored CPU, memory, and disk usage in real-time using the htop command.
- I also used the sar command to capture detailed system performance metrics over the 10-minute period, generating logs for **CPU** (cpu_usage.log), **memory** (memory_usage.log), and **disk** (disk_usage.log) usage.

Findings:

CPU Usage

- CPU usage remained **consistently above 80%** throughout the test.
- Analysis of *cpu_usage.log* showed most CPU cycles were spent in **user mode**, indicating the load was primarily application-driven (Logstash/Graylog).
- **The I/O wait time (%iowait) stayed low,** meaning the CPU was not sitting around waiting for the disk to catch up, finish reading or writing data. Instead, it was busy the whole time, working hard to process the logs being sent in.

Memory Usage

- Memory usage peaked quickly and remained **above 85%** throughout the duration of the test.

- While Linux naturally uses memory for caching, my analysis of memory_usage.log showed **low swap usage**, meaning the system had not yet resorted to using disk as extended memory.
- However, the tight memory margin indicated that **RAM resources were nearly saturated**, which would limit the ability to handle any additional load without risk of swapping or OOM (out-of-memory) errors.

Disk I/O Performance

- Disk activity logs from disk_usage.log showed **moderate but consistent write operations**, particularly on devices sda and dm-0.
- Average disk throughput during the test hovered around **2.4 MB/s write** and **1.9 MB/s read**.
- The %util metric (percentage of time the disk was busy) averaged around **15%**, and the **average wait time (await) remained below 1 millisecond**, which indicated **no I/O bottlenecks** were present under this workload.

Log Ingestion Time

- Despite the high CPU and memory usage, Graylog successfully **indexed all logs within 5 seconds**, meeting the desired ingestion latency benchmark for this baseline test.

Summary & Conclusion

Although the system technically handled the 500 EPS workload, it did so at **near-maximum capacity for CPU and memory**. This reveals a critical performance limitation: what was meant to be a baseline scenario already taxed the system significantly. Based on this, I originally anticipated the system could handle 500–1000 EPS, but these findings suggest **500 EPS is the upper bound before optimisation is necessary**.

The disk I/O performance remained stable, but the **high CPU and memory usage indicate this configuration is not production-ready** at higher volumes.

Challenges Encountered

Although I completed the baseline test, I encountered an issue shortly after starting the load testing. A few minutes into the test, the VM unexpectedly stopped running, and it showed the following errors:

- **BLKCACHE_IOERR**: This indicated that the I/O cache failed due to insufficient disk space. In response, I:
  - Enabled "Use Host I/O Cache" in VirtualBox.
  - Increased the VM's allocated storage from 50GB to 60GB.
- **DrvVD_DISKFULL**: This error revealed that the host machine itself had run out of disk space, not just the VM. To troubleshoot, I:
  - Further increased the VM storage to 70GB.

Despite these adjustments, the errors persisted. I eventually understood that the root cause couldn't be fixed with VM configuration, because it was the host machine's disk being completely full, which caused VirtualBox to pause the VM.

To resolve this:

- I freed up disk space on the host system by removing unnecessary files.
- I deleted the Ubuntu Desktop VM and installed a new, lighter Ubuntu Server image, which reduced resource consumption significantly.

These changes allowed me to resume testing without interruption and ensured that host device's limitations would no longer interfere with the performance testings.

Test Scenario 2: Stress Test - Increasing Log Volume

Steps Taken

- I injected 600, 800, and 1,000 EPS into Graylog for a period of 3 minutes each.
- During each stage, I used the *sar* command on the Ubuntu server to record CPU, memory, and disk usage every second over the 3-minute period (*sar -u 1 180 > cpu_usage.log & sar -r 1 180 > memory_usage.log & sar -d 1 180 > disk_usage.log*).
- I observed the Graylog Web UI during testing, monitoring key indicators like incoming vs processed message rates, disk journal usage, and system performance metrics.
- After each stage, I analysed the system logs and resource usage to identify signs of degradation or resource exhaustion.

Findings: 600 EPS Stage

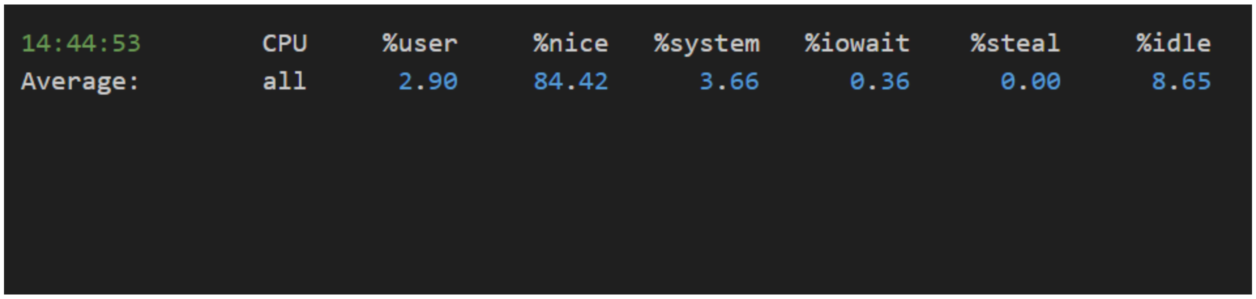
At the initial 600 EPS level, system resource usage was recorded and analysed:

CPU Usage

- **%idle** was low (on average about 8.65%), indicating the CPU was busy, but not fully saturated.
- **%iowait** was almost 0%, showing the CPU was not stalling while waiting for disk I/O.

At 600 EPS, the CPU was heavily utilised for log handling but not overwhelmed. There was no CPU bottleneck.

cpu_usage.log:



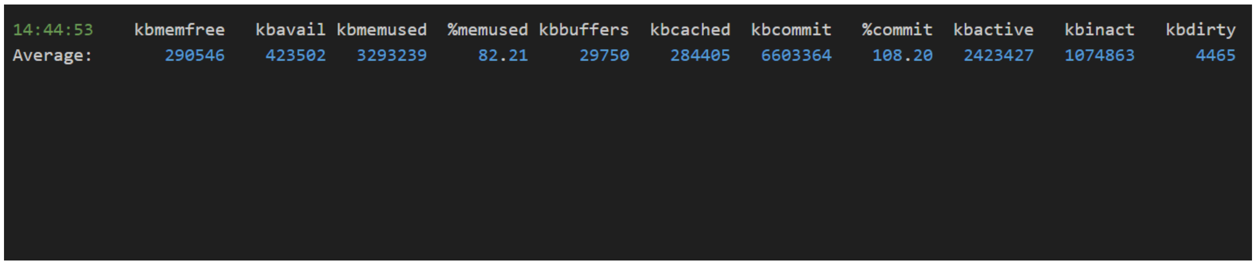
```
14:44:53        CPU     %user     %nice   %system   %iowait    %steal     %idle
Average:        all      2.90     84.42      3.66      0.36      0.00      8.65
```

Memory Usage

- **Memory usage** hovered around 82%–86%, with a gradual decrease in free memory but there were no significant drops.
- **Commit percentage** was above 100%, which is normal in Linux (includes virtual memory allocations).
- **No swapping or Out-Of-Memory (OOM) events** were observed.

The system handled the memory pressure. Even though there was High usage was expected, but there were no signs of memory exhaustion (like running out of memory) or instability. Everything was still within safe limits.

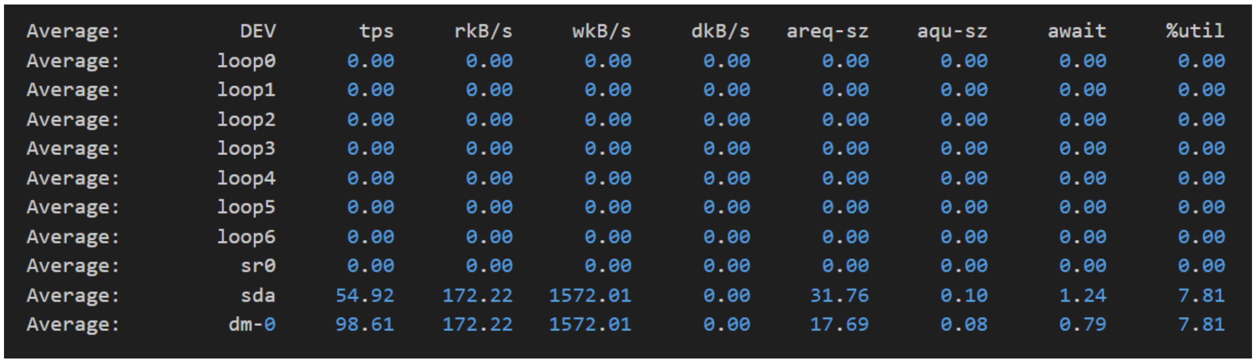
memory_usage.log


```
14:44:53    kbmemfree   kbavail kbmemused  %memused kbbuffers  kbcached  kbcommit   %commit  kbactive   kbinact   kbdirty
Average:       290546    423502   3293239     82.21     29750    284405   6603364    108.20   2423427   1074863      4465
```

Disk Usage

- **Disk utilisation** remained mostly below 10%, with periodic spikes during write bursts.
- **Disk I/O wait times** were low, suggesting no disk bottlenecks.
- **Disk journal usage** stayed below 0.5%, meaning Graylog could quickly process incoming logs without relying heavily on temporary disk storage. In Graylog, the **disk journal** acts as a temporary buffer when the system can't immediately process logs (like a queue). Low usage here shows Graylog was able to keep up with the log flow at 600 EPS without needing backup storage.

disk_usage.log


```
Average:          DEV       tps     rkB/s     wkB/s     dkB/s   areq-sz    aqu-sz     await     %util
Average:        loop0      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:        loop1      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:        loop2      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:        loop3      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:        loop4      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:        loop5      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:        loop6      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:          sr0      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:          sda     54.92    172.22   1572.01      0.00     31.76      0.10      1.24      7.81
Average:         dm-0     98.61    172.22   1572.01      0.00     17.69      0.08      0.79      7.81
```

Findings: 800 EPS Stage

During the 800 EPS test phase, I monitored system performance using both the Graylog Metrics UI and system logs. On one or two occasions, I observed a small difference between incoming and processed events, just a few messages behind, but it was minimal and temporary. This suggests that the system was beginning to feel the increased pressure but was still processing messages fast enough to recover quickly.

Disk journal utilisation rose slightly to around **0.77–0.78%** but remained comfortably below critical thresholds. Messages were appended briefly before being processed, showing that the system's buffering mechanism was working efficiently.

CPU Usage

- %user and %system stayed low overall, with some short spikes — notably, user time peaked at around 46% during brief moments.
- %idle hovered at roughly **8.76%**, suggesting the CPU was busy but not fully maxed out.
- %iowait was consistently low, indicating the CPU was not stalled by waiting for disk operations.

The CPU was under heavier load than during the 600 EPS test but still within tolerable limits. No signs of sustained bottlenecks were present.

Memory Usage

- %memused remained between **81–86%**, consistent with earlier tests and showing high but stable usage.
- There was no evidence of memory exhaustion, swapping, or OOM (Out of Memory) errors during the test.

Memory usage stayed high but steady. The system had enough space to avoid performance issues or crashing due to memory pressure.

Disk Usage

- On average the disk utilisation (%util) was **8.72%** but overall it stayed below **30%** for most of the test, with occasional spikes that quickly subsided.
- await values (I/O wait time) were low, which indicated fast and responsive disk operations.
- Disk journal usage slightly increased but still stayed under 1%, showing Graylog could keep up with buffering and message processing.

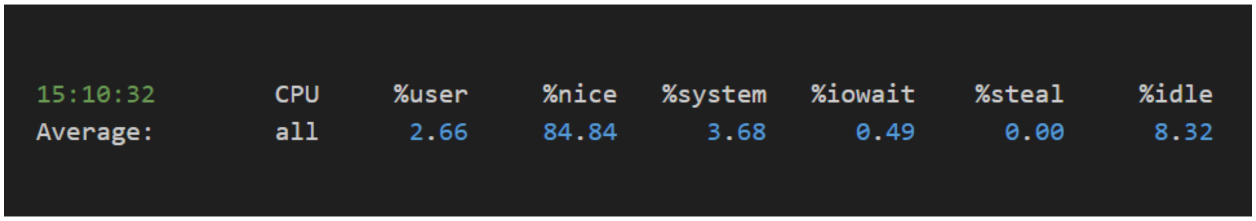
The disk handled the increased log volume effectively. No signs of performance degradation or I/O bottlenecks were observed at this level.

Findings: 1,000 EPS Stage

I conducted the test exactly the same way for 1,000 EPS and everything seemed fine. The only thing difference was a slight rise in disk disk utilisation, which had risen to about 1.50%.

CPU Usage

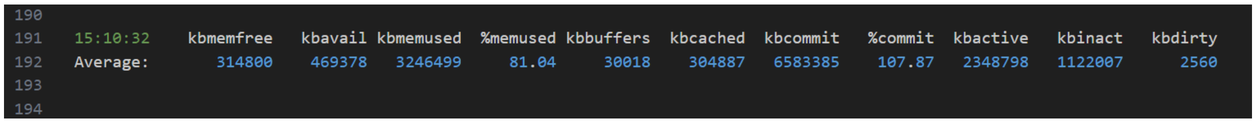
- **%user** and **%system** are low, and **%idle** is around 8%, so the CPU is not fully saturated.
    - **%iowait** is still low (0.49%), indicating little time waiting for disk I/O.
    - Occasional spikes in %user and drops in %nice, but no sustained CPU bottleneck.



```
15:10:32        CPU     %user     %nice   %system   %iowait    %steal     %idle
Average:        all      2.66     84.84      3.68      0.49      0.00      8.32
```

Memory Usage

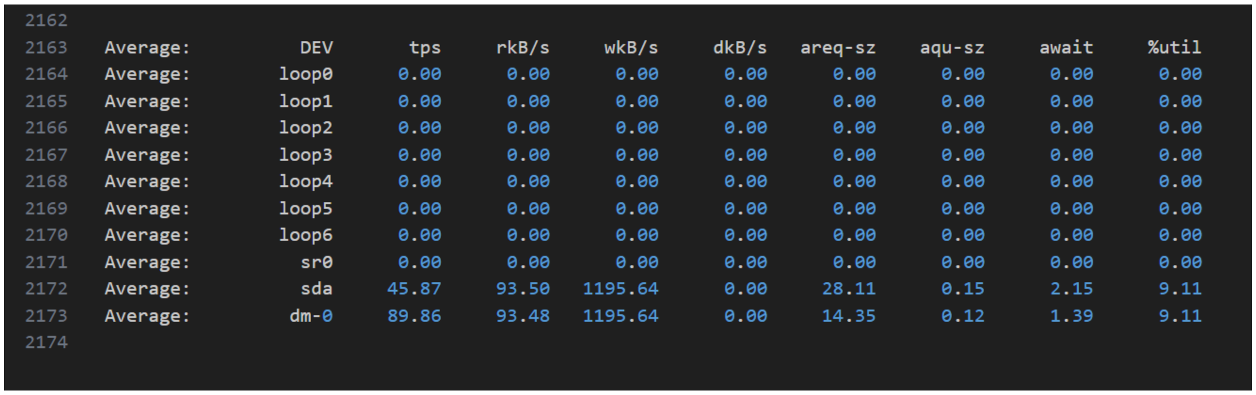
- **%memused** is around 81%, similar to previous tests, showing high but stable memory usage.



```
15:10:32    kbmemfree   kbavail kbmemused  %memused kbbuffers  kbcached  kbcommit   %commit  kbactive   kbinact   kbdirty
Average:       314800    469378   3246499     81.04     30018    304887   6583385    107.87   2348798   1122007      2560
```

Disk Usage

- **%util** (disk utilization) averages around 8.7%, with some spikes up to ~30%, but no sustained high usage.
- **await** (I/O wait time) is low, so disk is not at a bottleneck.
- The Disk handled the increased log volume without significant delays and **await** (I/O wait time) was low meaning it wasn't far from experiencing any bottlenecks.



```
Average:          DEV       tps     rkB/s     wkB/s     dkB/s   areq-sz    aqu-sz     await     %util
Average:        loop0      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:        loop1      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:        loop2      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:        loop3      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:        loop4      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:        loop5      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:        loop6      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:          sr0      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:          sda     45.87     93.50   1195.64      0.00     28.11      0.15      2.15      9.11
Average:         dm-0     89.86     93.48   1195.64      0.00     14.35      0.12      1.39      9.11
```

The system handled 1,000 EPS without resource exhaustion or instability. CPU, memory, and disk usage remain within healthy ranges, similar to lower EPS levels.

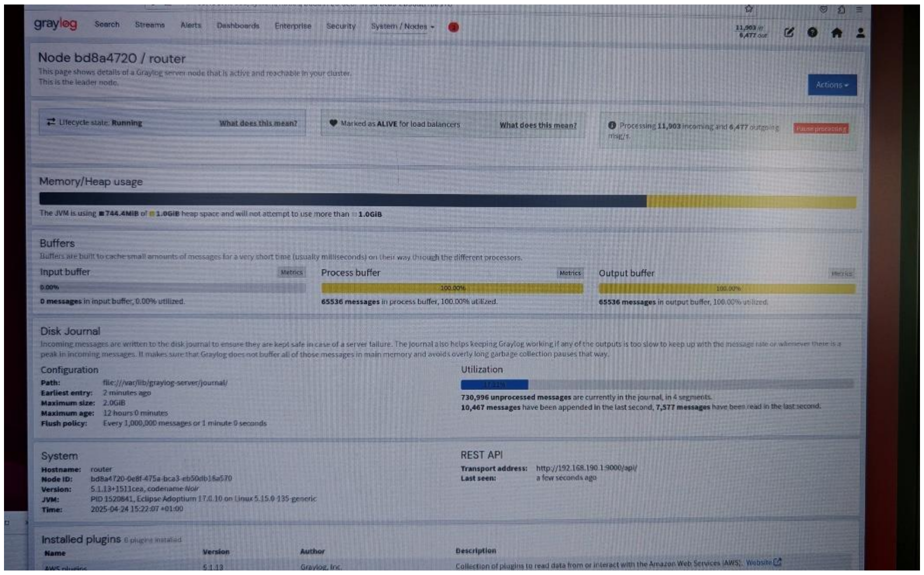
Findings: Infinite Log Input Test - Final Stress Level

After successfully testing at various EPS levels between our estimated system capacity range (500–1,000 EPS), and observing that the system remained mostly stable, albeit with high resource usage, I decided to push the system beyond its assumed threshold. The goal was to evaluate how Graylog would behave under continuous log ingestion, and the system acts when it reaches its breaking point.

To do this, I modified the Logstash configuration and set the count parameter to 0, which disabled the log limit, allowing Logstash to send logs indefinitely.

Not after strting the test, I saw the system stress indicators rapidly increase:

- Within just 5 minutes, the disk journal utilisation rose from **1% to 17%.**
- Both the **process buffer** and **output buffer** in the Graylog UI quickly reached and remained at **100%**.



Node bd8a4720 / router

This page shows details of a Graylog server node that is active and reachable in your cluster.
This is the leader node.

Lifecycle state: **Running** What does this mean?

Marked as **ALIVE** for load balancers What does this mean?

Processing **11,903** incoming and **6,477** outgoing msg/s.

Memory/Heap usage

The JVM is using **744.4MiB** of **1.0GiB** heap space and will not attempt to use more than **1.0GiB**

Buffers

Buffers are built to cache small amounts of messages for a very short time (usually milliseconds) on their way through the different processors.

Input buffer	Process buffer	Output buffer
0.00%	100.00%	100.00%
0 messages in input buffer, 0.00% utilized.	**65536 messages** in process buffer, 100.00% utilized.	**65536 messages** in output buffer, 100.00% utilized.

Disk Journal

Incoming messages are written to the disk journal to ensure they are kept safe in case of a server failure. The journal also helps keeping Graylog working if any of the outputs is too slow to keep up with the message rate or whenever there is a peak in incoming messages. It makes sure that Graylog does not buffer all of those messages in main memory and avoids overly long garbage collection pauses that way.

Configuration

Path:	file:///var/lib/graylog-server/journal/
Earliest entry:	2 minutes ago
Maximum size:	2.0GiB
Maximum age:	12 hours 0 minutes
Flush policy:	Every 1,000,000 messages or 1 minute 0 seconds

Utilization

17.12%

730,996 unprocessed messages are currently in the journal, in 4 segments.
10,467 messages have been appended in the last second, **7,577 messages** have been read in the last second.

System

Hostname:	router
Node ID:	bd8a4720-0e8f-475a-bca3-eb50db18a570
Version:	5.1.13+1511cea, codename *Noir*
JVM:	PID 1520841, Eclipse Adoptium 17.0.10 on Linux 5.15.0-135-generic
Time:	2025-04-24 15:22:07 +01:00

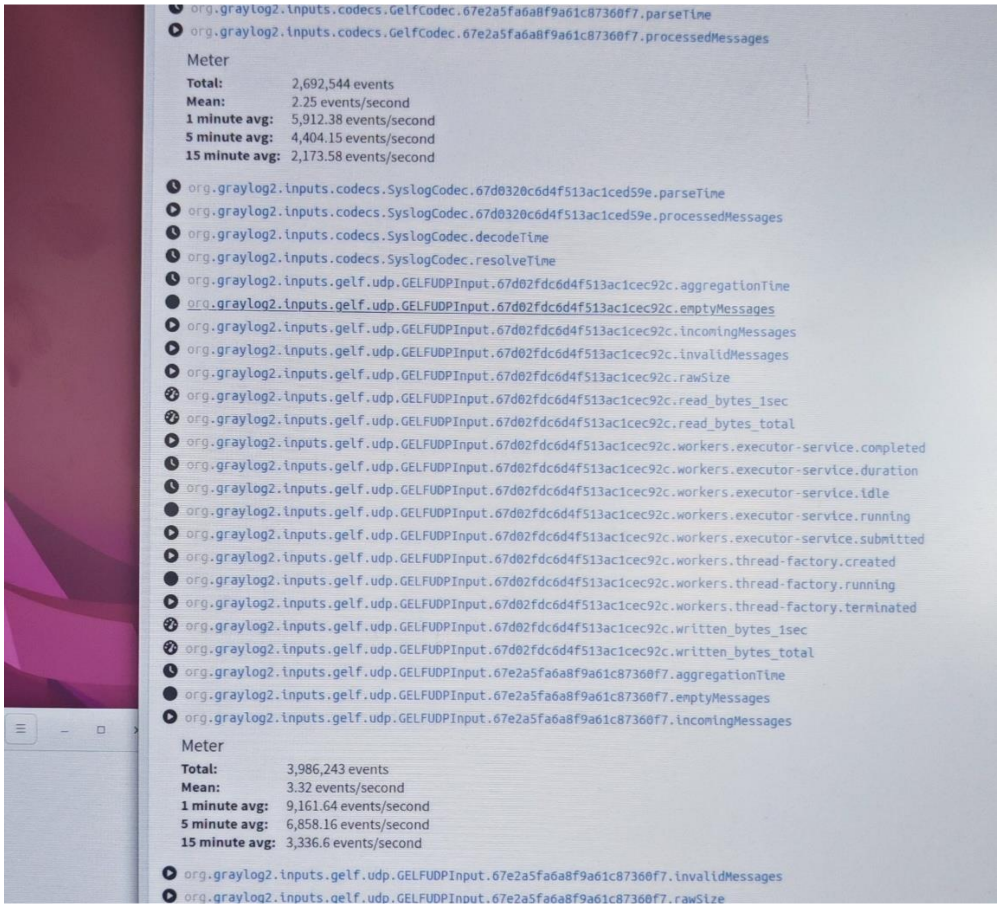
REST API

Transport address:	http://192.168.190.1:9000/api/
Last seen:	a few seconds ago

Installed plugins 8 plugins installed

Name	Version	Author	Description
AWS plugins	5.1.13	Graylog, Inc.	Collection of plugins to read data from or interact with the Amazon Web Services (AWS). Website

- The difference between **incoming and processed messages** began to widen significantly, highlighting that Graylog could no longer keep up with the ingest rate.



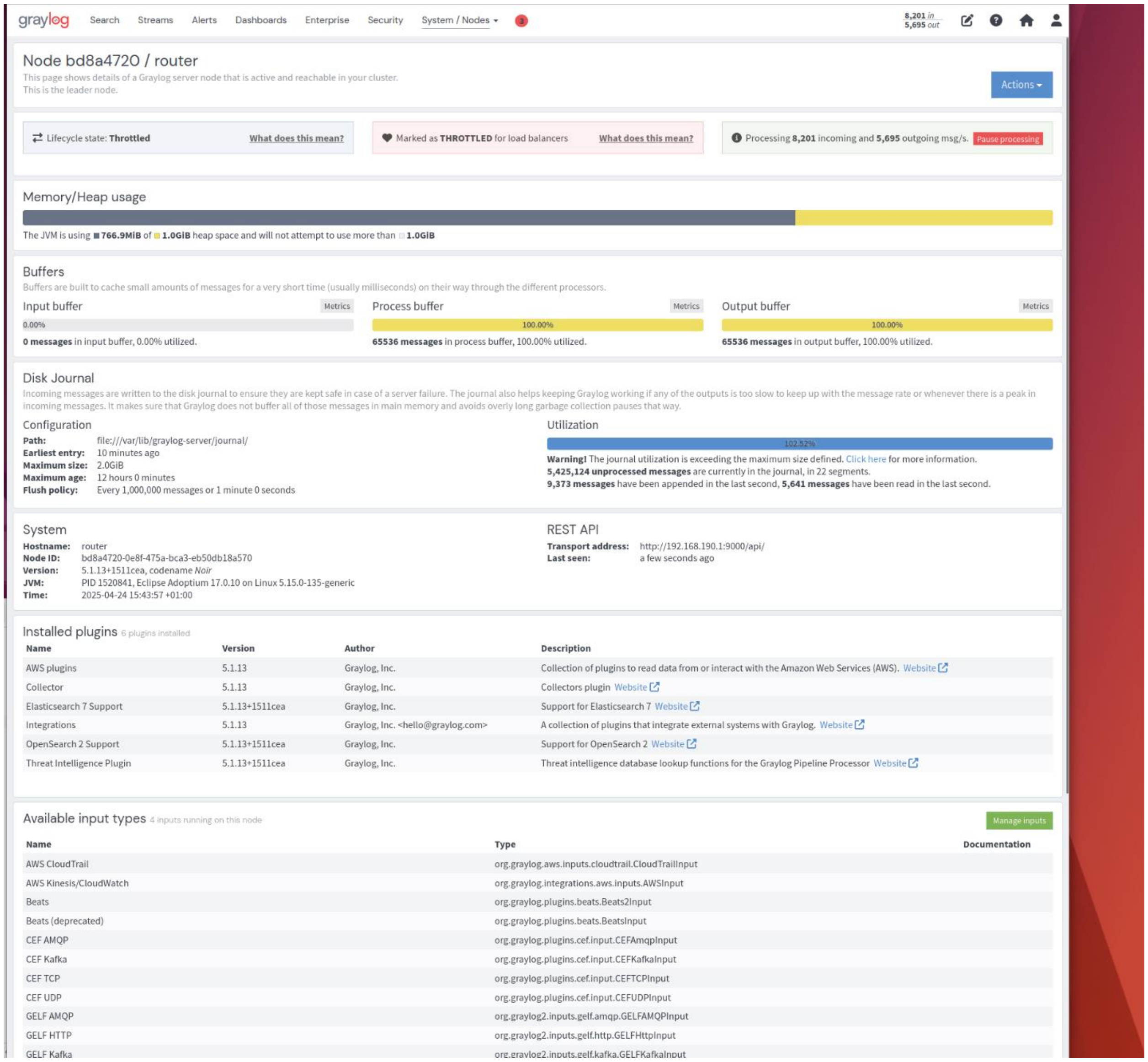
```
org.graylog2.inputs.codecs.GelfCodec.67e2a5fa6a8f9a61c87360f7.parseTime
org.graylog2.inputs.codecs.GelfCodec.67e2a5fa6a8f9a61c87360f7.processedMessages
  Meter
  Total:          2,692,544 events
  Mean:           2.25 events/second
  1 minute avg:   5,912.38 events/second
  5 minute avg:   4,404.15 events/second
  15 minute avg:  2,173.58 events/second

org.graylog2.inputs.codecs.SyslogCodec.67d0320c6d4f513ac1ced59e.parseTime
org.graylog2.inputs.codecs.SyslogCodec.67d0320c6d4f513ac1ced59e.processedMessages
org.graylog2.inputs.codecs.SyslogCodec.decodeTime
org.graylog2.inputs.codecs.SyslogCodec.resolveTime
org.graylog2.inputs.gelf.udp.GELFUDPInput.67d02fdc6d4f513ac1cec92c.aggregationTime
org.graylog2.inputs.gelf.udp.GELFUDPInput.67d02fdc6d4f513ac1cec92c.emptyMessages
org.graylog2.inputs.gelf.udp.GELFUDPInput.67d02fdc6d4f513ac1cec92c.incomingMessages
org.graylog2.inputs.gelf.udp.GELFUDPInput.67d02fdc6d4f513ac1cec92c.invalidMessages
org.graylog2.inputs.gelf.udp.GELFUDPInput.67d02fdc6d4f513ac1cec92c.rawSize
org.graylog2.inputs.gelf.udp.GELFUDPInput.67d02fdc6d4f513ac1cec92c.read_bytes_1sec
org.graylog2.inputs.gelf.udp.GELFUDPInput.67d02fdc6d4f513ac1cec92c.read_bytes_total
org.graylog2.inputs.gelf.udp.GELFUDPInput.67d02fdc6d4f513ac1cec92c.workers.executor-service.completed
org.graylog2.inputs.gelf.udp.GELFUDPInput.67d02fdc6d4f513ac1cec92c.workers.executor-service.duration
org.graylog2.inputs.gelf.udp.GELFUDPInput.67d02fdc6d4f513ac1cec92c.workers.executor-service.idle
org.graylog2.inputs.gelf.udp.GELFUDPInput.67d02fdc6d4f513ac1cec92c.workers.executor-service.running
org.graylog2.inputs.gelf.udp.GELFUDPInput.67d02fdc6d4f513ac1cec92c.workers.executor-service.submitted
org.graylog2.inputs.gelf.udp.GELFUDPInput.67d02fdc6d4f513ac1cec92c.workers.thread-factory.created
org.graylog2.inputs.gelf.udp.GELFUDPInput.67d02fdc6d4f513ac1cec92c.workers.thread-factory.running
org.graylog2.inputs.gelf.udp.GELFUDPInput.67d02fdc6d4f513ac1cec92c.workers.thread-factory.terminated
org.graylog2.inputs.gelf.udp.GELFUDPInput.67d02fdc6d4f513ac1cec92c.written_bytes_1sec
org.graylog2.inputs.gelf.udp.GELFUDPInput.67d02fdc6d4f513ac1cec92c.written_bytes_total
org.graylog2.inputs.gelf.udp.GELFUDPInput.67e2a5fa6a8f9a61c87360f7.aggregationTime
org.graylog2.inputs.gelf.udp.GELFUDPInput.67e2a5fa6a8f9a61c87360f7.emptyMessages
org.graylog2.inputs.gelf.udp.GELFUDPInput.67e2a5fa6a8f9a61c87360f7.incomingMessages
  Meter
  Total:          3,986,243 events
  Mean:           3.32 events/second
  1 minute avg:   9,161.64 events/second
  5 minute avg:   6,858.16 events/second
  15 minute avg:  3,336.6 events/second

org.graylog2.inputs.gelf.udp.GELFUDPInput.67e2a5fa6a8f9a61c87360f7.invalidMessages
org.graylog2.inputs.gelf.udp.GELFUDPInput.67e2a5fa6a8f9a61c87360f7.rawSize
```

As the test continued:

- Disk journal utilisation kept climbing and eventually exceeded **80%**, which is a critical warning. Shortly after, it surpassed **100%**, triggering a warning notification in the Graylog interface.



Node bd8a4720 / router

This page shows details of a Graylog server node that is active and reachable in your cluster.
This is the leader node.

Lifecycle state: **Throttled**

Marked as **THROTTLED** for load balancers

Processing **8,201** incoming and **5,695** outgoing msg/s.

Memory/Heap usage

The JVM is using **766.9MiB** of **1.0GiB** heap space and will not attempt to use more than **1.0GiB**

Buffers

Buffers are built to cache small amounts of messages for a very short time (usually milliseconds) on their way through the different processors.

Input buffer: 0.00% — **0 messages** in input buffer, 0.00% utilized.

Process buffer: 100.00% — **65536 messages** in process buffer, 100.00% utilized.

Output buffer: 100.00% — **65536 messages** in output buffer, 100.00% utilized.

Disk Journal

Incoming messages are written to the disk journal to ensure they are kept safe in case of a server failure. The journal also helps keeping Graylog working if any of the outputs is too slow to keep up with the message rate or whenever there is a peak in incoming messages. It makes sure that Graylog does not buffer all of those messages in main memory and avoids overly long garbage collection pauses that way.

Configuration

Path:	file:///var/lib/graylog-server/journal/
Earliest entry:	10 minutes ago
Maximum size:	2.0GiB
Maximum age:	12 hours 0 minutes
Flush policy:	Every 1,000,000 messages or 1 minute 0 seconds

Utilization

102.52%

Warning! The journal utilization is exceeding the maximum size defined. Click here for more information.
5,425,124 unprocessed messages are currently in the journal, in 22 segments.
9,373 messages have been appended in the last second, **5,641 messages** have been read in the last second.

System

Hostname:	router
Node ID:	bd8a4720-0e8f-475a-bca3-eb50db18a570
Version:	5.1.13+1511cea, codename *Noir*
JVM:	PID 1520841, Eclipse Adoptium 17.0.10 on Linux 5.15.0-135-generic
Time:	2025-04-24 15:43:57 +01:00

REST API

Transport address:	http://192.168.190.1:9000/api/
Last seen:	a few seconds ago

Installed plugins 6 plugins installed

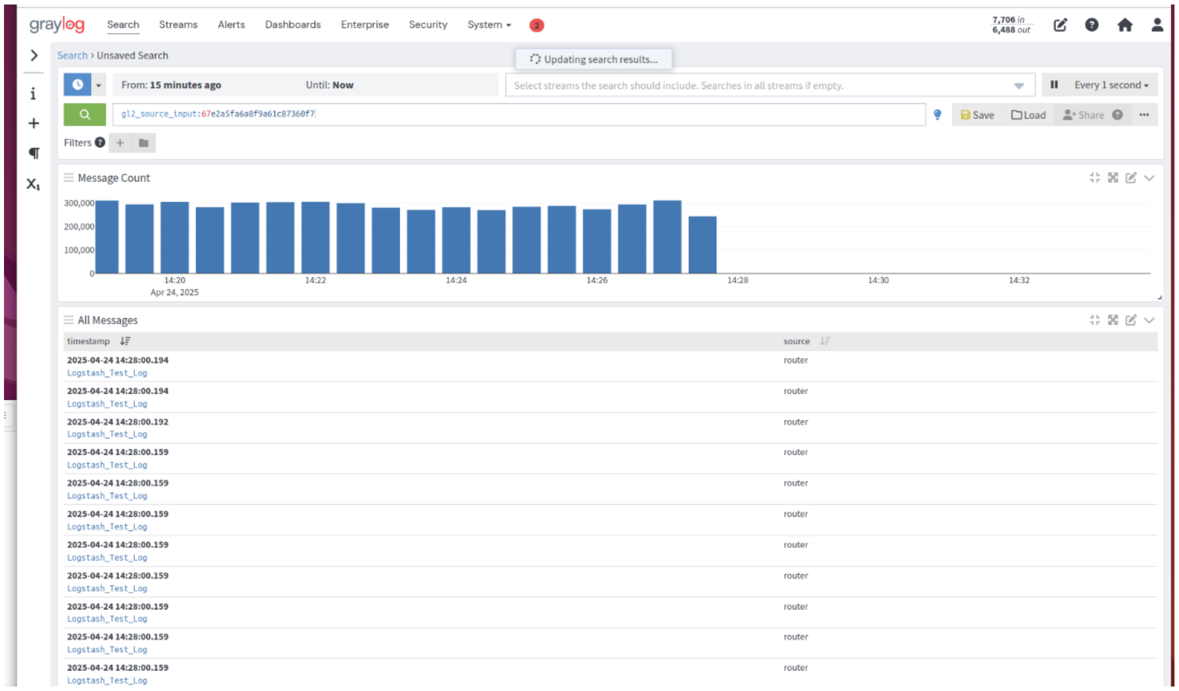
Name	Version	Author	Description
AWS plugins	5.1.13	Graylog, Inc.	Collection of plugins to read data from or interact with the Amazon Web Services (AWS). Website
Collector	5.1.13	Graylog, Inc.	Collectors plugin Website
Elasticsearch 7 Support	5.1.13+1511cea	Graylog, Inc.	Support for Elasticsearch 7 Website
Integrations	5.1.13	Graylog, Inc. <hello@graylog.com>	A collection of plugins that integrate external systems with Graylog. Website
OpenSearch 2 Support	5.1.13+1511cea	Graylog, Inc.	Support for OpenSearch 2 Website
Threat Intelligence Plugin	5.1.13+1511cea	Graylog, Inc.	Threat intelligence database lookup functions for the Graylog Pipeline Processor Website

Available input types 4 inputs running on this node

Name	Type	Documentation
AWS CloudTrail	org.graylog.aws.inputs.cloudtrail.CloudTrailInput	
AWS Kinesis/CloudWatch	org.graylog.integrations.aws.inputs.AWSInput	
Beats	org.graylog.plugins.beats.Beats2Input	
Beats (deprecated)	org.graylog.plugins.beats.BeatsInput	
CEF AMQP	org.graylog.plugins.cef.input.CEFAmqpInput	
CEF Kafka	org.graylog.plugins.cef.input.CEFKafkaInput	
CEF TCP	org.graylog.plugins.cef.input.CEFTCPInput	
CEF UDP	org.graylog.plugins.cef.input.CEFUDPInput	
GELF AMQP	org.graylog2.inputs.gelf.amqp.GELFAMQPInput	
GELF HTTP	org.graylog2.inputs.gelf.http.GELFHttpInput	
GELF Kafka	org.graylog2.inputs.gelf.kafka.GELFKafkaInput	

The Graylog UI displayed a node status of **"Lifecycle State: Throttled"** and **"Marked as THROTTLED for load balancers"**, confirming that the system was under excessive load. This internal throttling mechanism is Graylog’s way of reducing input pressure to maintain stability, further validating that the system had reached, and exceeded, its sustainable performance limits under current configurations.

- The **query response time** noticeably degraded as well. This was evident in the UI, where the message histogram (bar graph) that updates every second started behaving irregularly, bars for timestamps that had already passed were still rising, showing delayed message visibility and indicating backend lag.



```
gl2_source_input:67e2a5fa6a8f9a61c87360f7
```

timestamp	source
2025-04-24 14:28:00.194 Logstash_Test_Log	router
2025-04-24 14:28:00.194 Logstash_Test_Log	router
2025-04-24 14:28:00.192 Logstash_Test_Log	router
2025-04-24 14:28:00.159 Logstash_Test_Log	router
2025-04-24 14:28:00.159 Logstash_Test_Log	router
2025-04-24 14:28:00.159 Logstash_Test_Log	router
2025-04-24 14:28:00.159 Logstash_Test_Log	router
2025-04-24 14:28:00.159 Logstash_Test_Log	router
2025-04-24 14:28:00.159 Logstash_Test_Log	router
2025-04-24 14:28:00.159 Logstash_Test_Log	router
2025-04-24 14:28:00.159 Logstash_Test_Log	router

- The gap between **incoming and processed logs** had ballooned to several hundred thousand messages, with the system clearly unable to keep up with the input rate. This confirmed that the system had crossed its sustainable performance boundary.

This infinite input test highlighted the system's absolute upper limit. Despite previous stability up to 1,000 EPS, the unbounded log stream caused significant performance degradation in less than 15 minutes. The growing backlog in the disk journal, 100% buffer saturation, and delayed UI updates confirmed that the Graylog instance had reached, and exceeded, its processing capacity.

Overall Insights – Test Scenario 2

While the system appeared to handle log ingestion rates between 500 and 1,000 EPS during controlled testing, I believe the results may not fully reflect how the system would perform under realistic conditions. The test logs I generated were relatively lightweight and uniform in structure, meaning they didn't reflect the full complexity or volume of data that would likely be present in logs generated by Catnip Games' actual infrastructure.

Given that I was already observing consistently high CPU and memory usage as early as 500 EPS, it's reasonable to infer that heavier, more complex log entries could lead to earlier system saturation or degraded performance at even lower EPS thresholds.

Therefore, while the system demonstrated relative stability during this stress test, the results suggest limited headroom for scalability. Any increase in log complexity, indexing demands, or additional features could push the system beyond its sustainable limits.

Test Scenario 3: Query Performance Under Load

Steps Taken:

- I injected 1,000 EPS into Graylog for a continuous period of 10 minutes to simulate a high-load environment.
- During this time, I executed multiple queries via the Graylog web interface, including:
  - Keyword-based searches (e.g., failed login, root access).
  - Time-range filters (e.g., logs from the last 10 minutes).
  - Source-specific queries (e.g., source:logstash)
- I recorded the query response times under load and compared them against baseline established during earlier low-load conditions.

Findings:

- Baseline Query Response Time:
  - Under idle or minimal load, Graylog consistently returned search results within 2–3 seconds.
- Query Response at 500–800 EPS:
  - Response time slightly increased but generally stayed within 3–5 seconds.
  - Queries remained usable and responsive, though performance was near the upper limit of acceptable delay.
- Query Response at 1,000 EPS:

    - Despite the elevated log ingestion rate, query times remained stable and were still under 5 seconds.
    - There was no major degradation or timeout, and the interface remained responsive during searches.

Query performance remained within acceptable limits throughout all test stages, including 1,000 EPS. While response times increased slightly as the system load rose, Graylog was still able to process and return results reliably. This suggests that under the current configuration, the system can maintain effective user-level interaction even under elevated throughput, though it is approaching its threshold.

However, it's important to note that this test was only conducted after allowing the system to recover from the Infinite Log Input Test. That prior test had severely overwhelmed system resources, causing the disk journal, output buffer, and process buffer to saturate and degrade performance. Attempting query testing in that state would not have yielded valid or meaningful results. As such, I paused the test environment, stopped Logstash, and waited approximately 30 minutes for system metrics (disk utilisation, CPU load, etc.) to return to baseline before starting Test Scenario 3.

Recommendations

Based on the performance testing conducted across three scenarios (Baseline Performance Test, Incremental Stress Testing, and Query Performance Under Load), I propose the following recommendations to improve the stability, scalability, and operational readiness of the Graylog SIEM environment:

1. Scale System Resources for Production Use

While the system handled up to 1,000 EPS, CPU and memory usage were consistently high starting from 500 EPS, suggesting limited headroom.

To support production-level log volumes, upgrading system specifications, particularly:

- Increase RAM to handle large buffers and indexing operations.
- Use multi-node architecture (dedicated Graylog, Elasticsearch, and MongoDB nodes).

2. Implement Ingestion Controls

The infinite log input test demonstrated that unbounded ingestion leads to disk saturation and processing lag.

Introduce log throttling to limit the rate at which logs are sent to the system (e.g., capped EPS per source) at input to prevent system overload during log spikes, stress periods.

3. Set Up Monitoring & Alerting for Critical Metrics

Monitor key indicators such as:

- Process/output buffer usage
- Disk journal utilisation
- Difference between incoming and processed messages
- System resource utilisation

Configure alerting in Graylog or via external tools to notify administrators before the system reaches critical thresholds.

4. Reassess Log Retention and Rotation Policies

Index rotation was configured effectively for testing (daily rotation, 7-day retention), but a production environment may require:

- Granular rotation based on size or time.
- Archive strategies for long-term storage (e.g., cold storage, S3, Glacier).
- Periodic review of retention rules to balance performance and compliance.

Conclusion

The performance testing demonstrated that the single-node Graylog SIEM environment is capable of handling up to 1,000 Events Per Second (EPS) without critical failure. However, at that level, system resources are often at high usage levels which indicate that it won’t take much for the system to reach its processing limits.

Overall, while the prototype SIEM setup was fine for testing, it is not yet suitable for long-term or enterprise production use in its current form. To move forward, the system will require hardware upgrades, architectural adjustments (multi-node deployment), and improved ingestion controls. With these improvements in place, Graylog could be scaled confidently to support Catnip Games International’s evolving security monitoring needs.