Resource Usage Graph Metric Definitions

The **Resource Usage** graph provides information about the system resources consumed during test execution. These metrics help identify potential resource constraints and correlate resource usage with test performance.

The metrics are divided into two categories:

- **JVM Metrics** — Metrics related to the Java Virtual Machine (JVM) running the Appvance process.
- **OS Metrics** — Metrics reported by the operating system for the machine where the test is running.

JVM Metrics

1. Heap Usage

Heap Usage shows the amount of Java heap memory currently being used by the JVM.

The Java heap is a runtime memory area used by the JVM to store objects and arrays created during application execution. For example, test execution may create objects to process requests, responses, variables, and other runtime data.

Appvance initializes the JVM heap with a default maximum heap size of **1500 MB**, according to the configuration documented for this version.

Heap usage can increase as the test creates more objects. The JVM's Garbage Collector can subsequently reclaim memory occupied by objects that are no longer being used.

For example:

```
Heap Usage


400 MB  →  700 MB  →  1100 MB  →  Garbage Collection  →  600 MB
```

An increase in heap usage does not necessarily indicate a problem. When analyzing this metric, look at its behavior over time.

If heap usage continuously increases and approaches the maximum heap size without being reclaimed, this may indicate memory pressure and should be investigated.

> **Note:** The default heap size can vary depending on the Appvance version and configuration. Verify the JVM configuration for the specific environment being tested rather than assuming that 1500 MB applies to every installation.

2. Garbage Collector

The **Garbage Collector (GC)** is a JVM process responsible for identifying and reclaiming memory occupied by objects that are no longer required.

The Resource Usage graph reports the CPU usage associated with the Garbage Collector.

For example:

```
Application creates objects
        ↓
Heap usage increases
        ↓
Garbage Collector runs
        ↓
Unused objects are reclaimed
        ↓
Available heap memory increases
```

Higher Garbage Collector CPU usage can occur when the JVM needs to perform garbage collection more frequently or for longer periods.

A sustained increase in Garbage Collector activity, particularly when accompanied by increasing heap usage, high CPU usage, or increasing response times, may indicate memory pressure.

3. Thread Count

Thread Count shows the current number of active threads in the JVM.

A thread is an execution unit within a process. The JVM can use multiple threads to perform work concurrently.

Thread count can increase as the amount of concurrent work increases. However, a continuously increasing or unusually high thread count may require investigation, especially when CPU or memory usage is also high.

For example:

```
Thread Count


100  →  200  →  400  →  800
```

Thread count should be evaluated together with the workload and other resource metrics. A high thread count alone does not necessarily indicate a problem.

OS Metrics

1. CPU Usage

CPU Usage shows CPU utilization as reported by the operating system.

CPU usage indicates how much processing capacity is being consumed by the system during test execution.

For example:

```
CPU Usage


30%  →  50%  →  75%  →  95%
```

Sustained high CPU utilization can indicate that the machine is approaching its processing capacity.

If CPU usage remains close to maximum while response times increase or throughput stops increasing, CPU may be a resource bottleneck.

2. MEM

MEM shows the percentage of system memory being used relative to the total memory available, as reported by the operating system.

This metric represents system-level memory usage and should not be confused with **Heap Usage**.

For example, a machine with 16 GB of RAM using approximately 8 GB would report approximately 50% memory utilization.

The distinction is important:

```
Heap Usage
    ↓
Memory used by the JVM heap


MEM
    ↓
System-level memory usage reported by the OS
```

System memory usage can include memory used by the JVM, other processes, the operating system, and other memory consumers.

Sustained high memory utilization may indicate memory pressure and should be investigated together with Heap Usage and Garbage Collector activity.

3. NET

NET shows network traffic as reported by the operating system.

Network traffic represents data being transferred by the machine during test execution.

For example, test activity can generate network traffic when:

- Requests are sent to the application.
- Responses are received from the application.
- Large files or responses are transferred.
- Multiple virtual users generate concurrent network activity.

An increase in network traffic is not necessarily a problem. The metric should be evaluated against the expected workload and the available network capacity.

Sustained high network utilization combined with increasing response times or reduced throughput may indicate a potential network bottleneck.

Interpreting Resource Usage Metrics

Resource Usage metrics should be analyzed together rather than individually.

For example:

```
CPU ↑
Heap ↑
Garbage Collector ↑
Response Time ↑
```

This combination may indicate that the JVM and system are experiencing increased resource pressure.

Another example:

```
CPU ≈ 100%
Heap stable
MEM normal
NET normal
Response Time ↑
```

In this case, CPU saturation may be a potential bottleneck.

Similarly:

```
Heap ↑
Garbage Collector ↑
MEM ↑
Response Time ↑
```

may indicate increasing memory pressure.

The Resource Usage graph is therefore most useful when its metrics are correlated with other test results, such as:

- Response time
- Throughput
- Error rate

- Virtual user load
- Request volume

Summary

Category	Metric	Description
JVM	**Heap Usage**	Amount of Java heap memory currently being used
JVM	**Garbage Collector**	CPU usage associated with JVM garbage collection
JVM	**Thread Count**	Current number of active JVM threads
OS	**CPU Usage**	CPU utilization reported by the operating system
OS	**MEM**	Percentage of system memory being used
OS	**NET**	Network traffic reported by the operating system

When investigating performance issues, use these metrics to determine whether the test environment may be approaching a **CPU, memory, JVM, thread, or network resource limit**.

Last updated: August 28, 2026

If you have any feedback on this topic, please contact us at documentation@appvance.ai.

Revision #3
Created 2026-07-31 15:38:10 UTC by Admin
Updated 2026-08-28 19:47:31 UTC by mmora