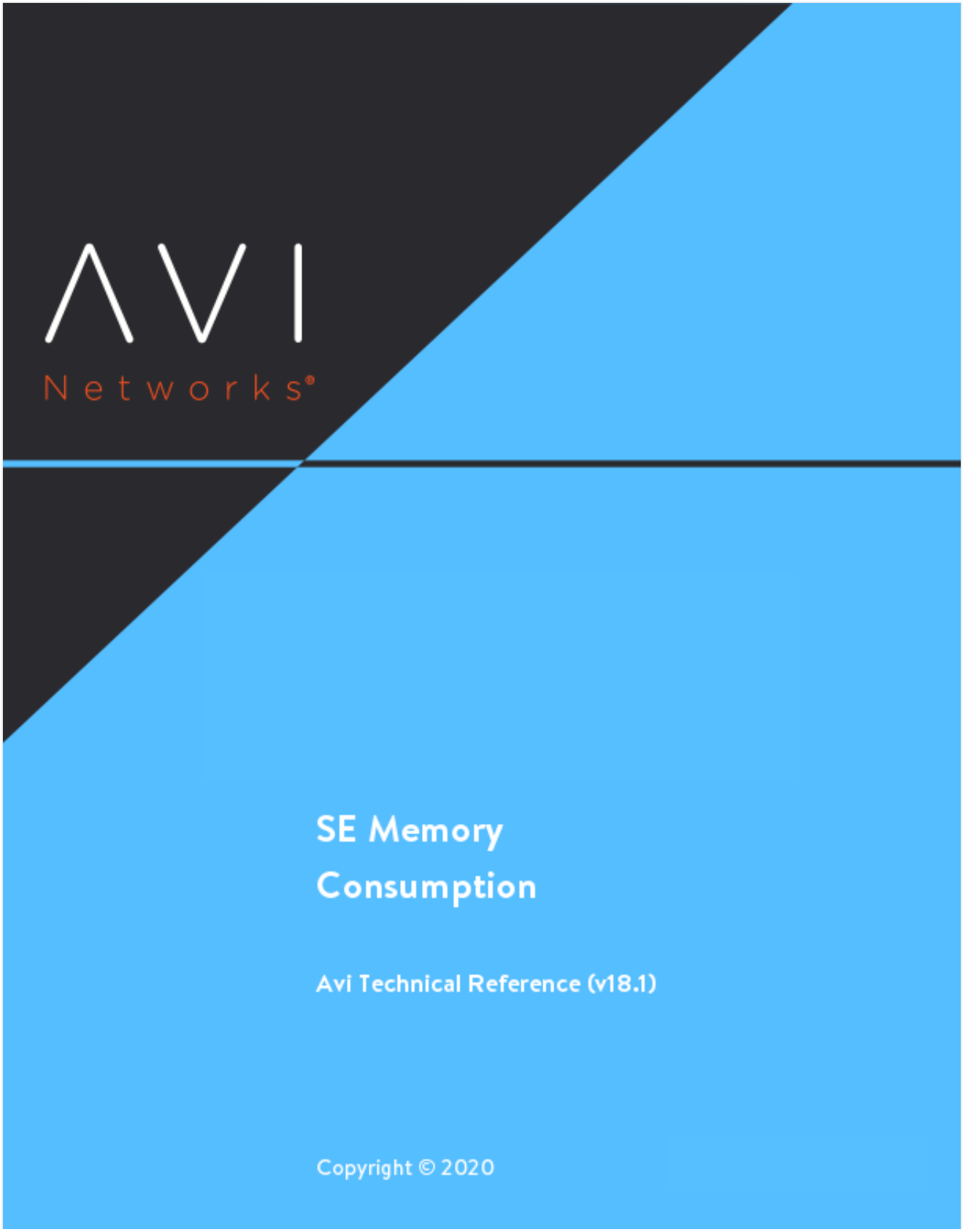



# SE Memory Consumption

Avi Technical Reference (v18.1)

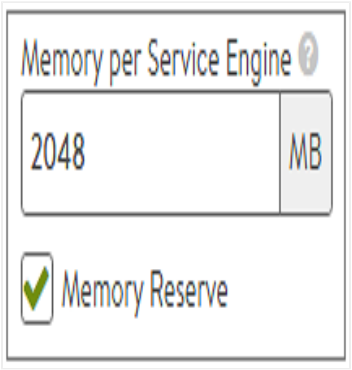
# SE Memory Consumption

view online

## Overview

This guide explains about calculating the utilization of memory within a Service Engine (SE) to estimate the number of concurrent connections or the amount of memory that may be allocated to features such as HTTP caching.

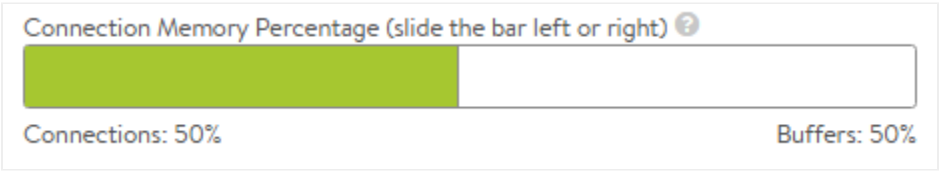
Service Engines support 1-128 GB memory. Avi Vantage's minimum recommendation is 2 GB. Providing more memory greatly increases the scale of capacity, as does adjusting the priorities for memory between concurrent connections and optimized performance buffers.




Memory allocation for Avi Vantage SE deployments in write access mode is configured via Infrastructure > Cloud > SE Group Properties. Changes to the Memory per Service Engine property only impact newly created SEs. For read or no access modes, the memory is configured on the remote orchestrator such as vCenter. Changes to existing SEs require the SE to be powered down prior to the change. ## Memory Allocation

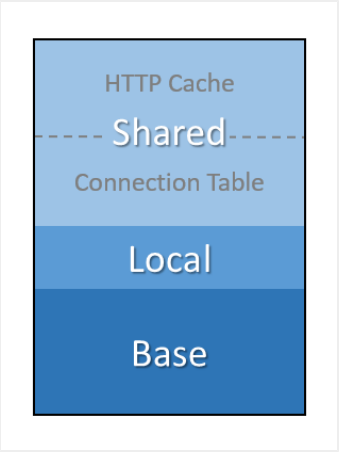
The following are the Service Engine's memory allocation:

| | | |
|---|---|---|
| Base | 500 MB | Required to turn on the SE (Linux plus basic SE functionality) |
| Local | 100 MB / core | Memory allocated per vCPU core |
| Shared | Remaining | Remaining memory is split between Connections and HTTP Cache |




The shared memory pool is divided between two components, namely, Connections and Buffers. A minimum of 10% must be allocated to each. Changing the Connection Memory Percentage slider will impact newly created SEs but will not impact the existing SEs.

Connections consists of the TCP, HTTP, and SSL connection tables. Memory allocated to connections directly impacts the total concurrent connections that a Service Engine can maintain.




Buffers consists of application layer packet buffers. These buffers are used for layer 4 through 7 to queue packets to provide improved network performance. For instance, if a client is connected to the Avi SE at 1mb/s with large latency and the server is connected to the SE at no latency and 10gb/s throughput, the server can respond to client queries by transmitting the entire response and move on to service the next client request. The SE will buffer the response and transmit it to the client at the much reduced speed, handling any retransmissions without needing to interrupt the server. This memory allocation also includes application centric features such as HTTP caching and improved compression.

Buffers maximize the number of concurrent connections by changing the priority towards Connections. Avi Vantage's calculations are based on the default setting, which is 50% of the shared memory available for connections.

### Concurrent Connections

Most Application Delivery Controller (ADC) benchmark numbers are based on an equivalent Transmission Control Path (TCP) Fast Path, which uses a simple memory table of client IP:port mapped to server IP:port. This uses very little memory, enabling extremely large concurrent connection numbers. However, it is also not relevant to the vast majority of real world deployments which rely on TCP and application layer proxying. Avi Vantage's benchmark numbers are based on full TCP proxy (L4), TCP plus HTTP proxy with buffering and basic caching plus DataScript (L7), and the same scenario with Transfer Layer Security Protocol (TLS) 1.2 between client and Avi Vantage.

The memory consumption numbers per connection listed below can be higher or lower. For instance, typical buffered HTTP request headers consume 2k, but they can be as high as 48k. The numbers below are intended to provide real world sizing guidelines.

Memory consumption per connection:

- 10 KB L4
- 20 KB L7
- 40 KB L7 + SSL (RSA or ECC)

To calculate the potential concurrent connections for a Service Engine, use the following formula:

Concurrent L4 connections = ((SE memory - 500 MB - (100 MB * num of vCPU)) * Connection Percent) / Memory per Connection

To calculate the potential concurrent connections for a Service Engine, use the following formula:

Concurrent L4 connections = ((SE memory - 500 MB - (100 MB * num of vCPU)) * Connection Percent) / Memory per Connection

To calculate layer 4 sessions (memory per connection = 10KB = 0.01MB) for an SE with 8 vCPU cores and 8 GB RAM, using a Connection Percentage of 50%, the math looks like: ((8000 - 500 - ( 100 * 8 )) * 0.50) / 0.01 = 3,35,000.

| | 1 vCPU | 4 vCPU | 32 vCPU |
|---|---|---|---|
| 1 GB | 36k | n/a | n/a |
| 4 GB | 306k | 279k | n/a |
| 32 GB | 2.82m | 2.80m | 2.52m |

The table above shows the number of concurrent connections for L4 (TCP Proxy mode) optimized for connections. ### View Allocation via CLI

From the CLI: `show serviceengine <SE Name> memdist` This command shows a truncated breakdown of memory distribution for the SE. This SE has one vCPU core with 141 MB allocated for the shared memory's connection table. The 'huge_pages' value of 91 means there are 91 pages of 2 MB each. This indicates 182 MB has been allocated for the shared memory's HTTP cache table.

```
[admin:controller]: > show serviceengine Avi-se-ktiwo memdist
+------------------------+--------------------------+
| Field                  | Value                    |
+------------------------+--------------------------+
| se_ref                 | Avi-se-bajip:se-0068b1   |
| huge_pages             | 91                       |
| conn_memory_mb         | 141                      |
| conn_memory_mb_per_core | 141                     |
+------------------------+--------------------------+
```

## View Allocation via API

The total memory allocated to the connection table and the percentage in use may be viewed. Use the following commands to query the API:

`https://<IP Address>/api/analytics/metrics/serviceengine/se-<SE UUID>?metric_id=se_stats.max_connection_mem_total` Returns the total memory available to the connection table. In the response snippet below, 141 MB is allocated.

```
"statistics": {
   "max": 141,
}
```

`https://<IP Address>/api/analytics/metrics/serviceengine/se-<SE UUID>?metric_id=se_stats.avg_connection_mem_usage&step=5`

Returns the average percent of memory used during the queried time period. In the result snippet below, 5% of the memory was in use.

```
"statistics": {
   "min": 5,
   "max": 5,
```

```
    "mean": 5
},
```