Memory usage in spark falls under two broad categories....

1) Execution memory - memory required for computations in shuffle like joins, sorts, aggregations...
2) Storage memory - is used for cache

In spark Execution memory and Storage memory share a common region.

when no Execution is happening... then your storage can acquire all the available memory and vice versa.....

Execution may Evict storage if necessary

2gb common unified region

2gb for storage

→ but this Eviction can happen until total storage memory usage falls unser a certain threshold

2gb if ~~both~~ all gb is used for storage...

→ now if some Execution / computations are coming they cannot Evict the Entire 2gb...

→ There is a certain threshold beyound which the Execution cannot Evict the storage..

→ Execution can Evict Storage upto a certain threshold....

Note → but Storage can not Evict Execution.

This design Ensures several desirable properties :-

1. Application which do not use caching can use Entire space for Execution.
2. Application that do not use caching can reserve a minimum Storage space.....

This makes your data blocks immune from being Evicted...

-- Executer - memory 4g

If you request a container/ Executer of 4GiB size ...
then you are actually requesting

4 GiB (heap memory)
+
max (384mb, 7% of 4GiB) (off heap memory) - overhead

4096 mb (Java heap)
+
384 mb (off heap) - overhead

The heap memory is sub divided ....
out of 4gb heap memory 300 mb is reserved ...

3.7gb

60% is for (storage & Execution) ~ 2.3gb

40% is for user memory - 1.4gb (to hold for user data structures, Spark related metadata)

out of 2.3gb 50% is the threshold for storage memory

This means we can cache data upto 1.5gb roughly without worrying about Eviction by Executions/ computations.