Executor Memory Architecture

Objective

Overview of the executor memory layout

Prerequisite for caching

Configuration tuning

Executor Memory Architecture

2 kinds of memory allocated to executors

- execution: joins, sorts, groups - usually for shuffles
- storage: caching RDDs in memory for reuse and fast retrieval

From Spark 1.6 memory is unified and shared between storage/execution

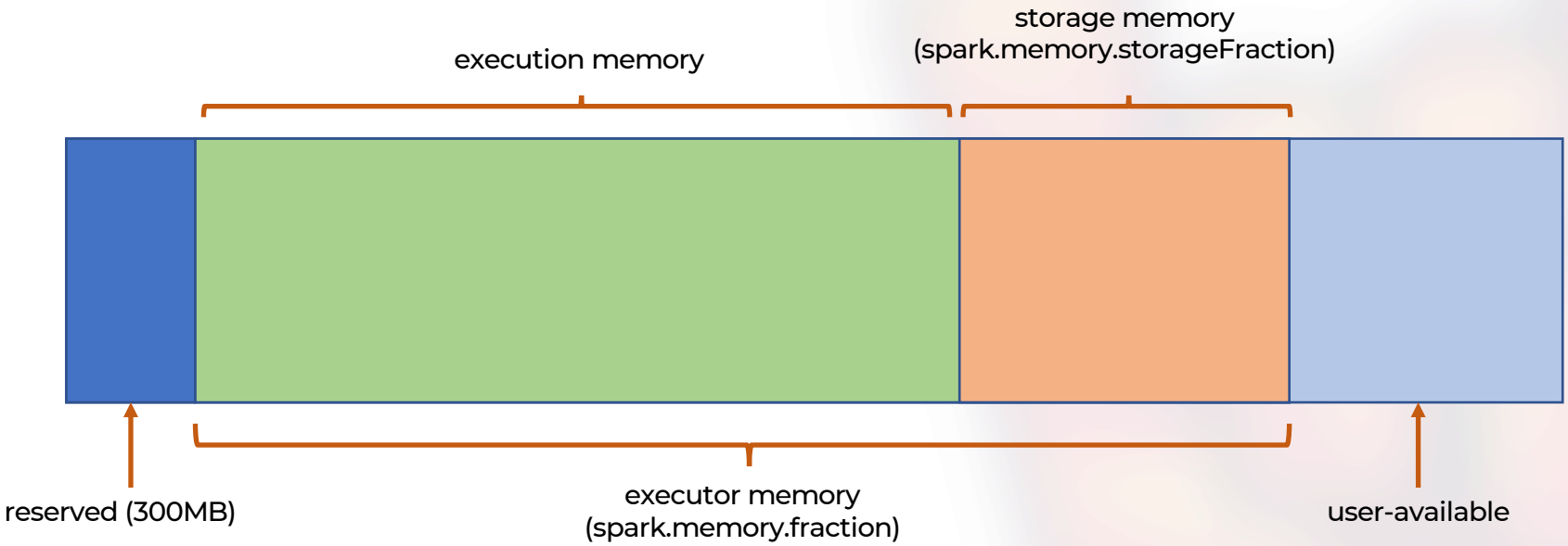
Storage memory can be used for execution

- LRU algorithm used to evict unused cached RDDs from memory

Fractions of storage vs execution can be configured at application start

- 300MB reserved
- spark.memory.fraction = executor-reserved memory (0 to 1)
- spark.memory.storageFraction = which memory ratio from the executor is reserved for storage (0 to 1)
- rest is available to user e.g. locally-allocated collections etc

Executor Memory Architecture

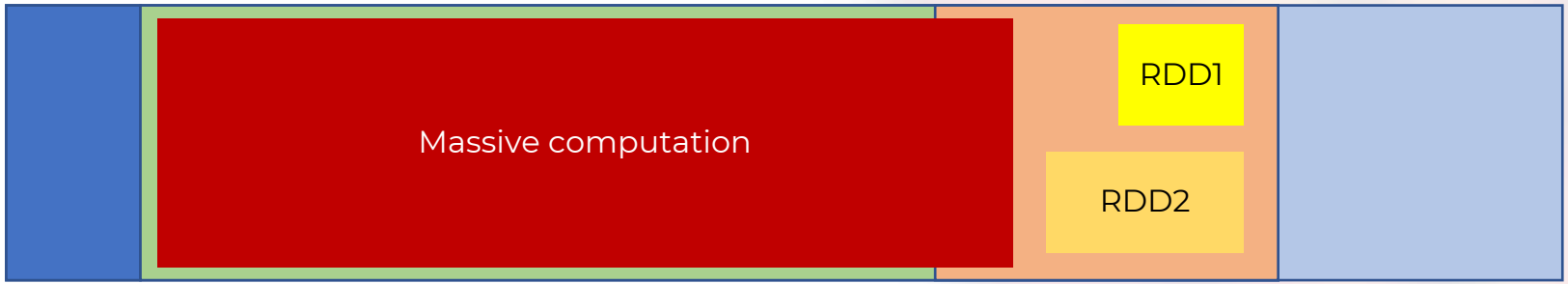


Executor Memory Architecture

Example 1: assume we have 2 RDDs cached in the Storage area

Executor Memory Architecture

Example 1: assume we have 2 RDDs cached in the Storage area

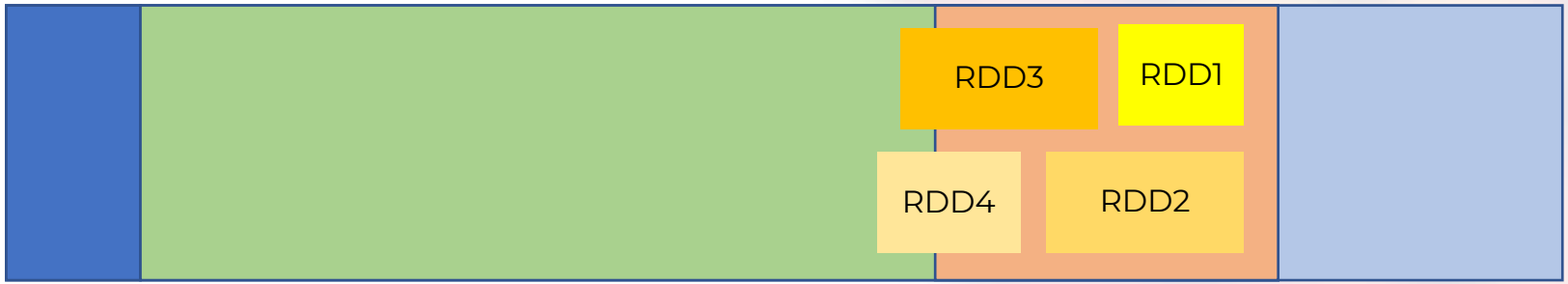


A massive computation (e.g. shuffle) can use the unused Storage if needed

Executor Memory Architecture

Example 2: assume we have many RDDs cached, overflowing Storage

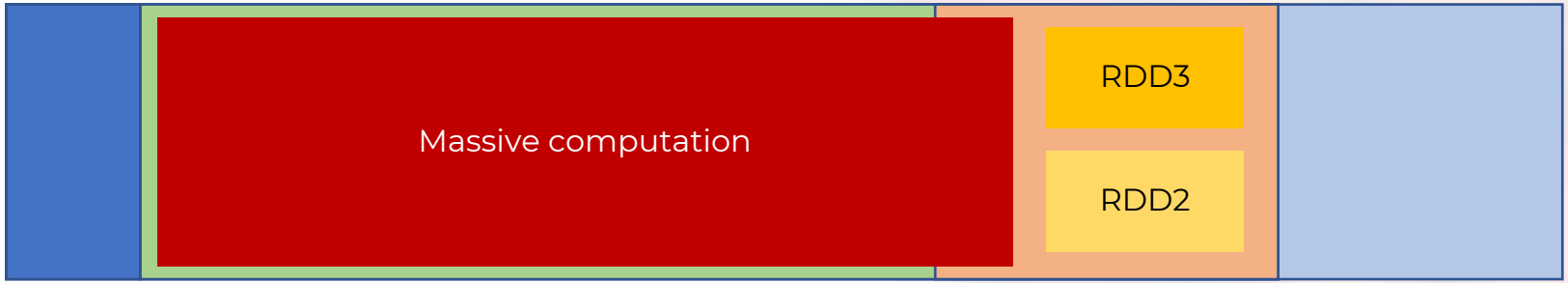
- it's possible if the compute zone is available
- a big computation would need to evict the LRU RDDs until at least the green area is free
- assume RDD1 and RDD4 are LRU



Executor Memory Architecture

Example 2: assume we have many RDDs cached, overflowing Storage

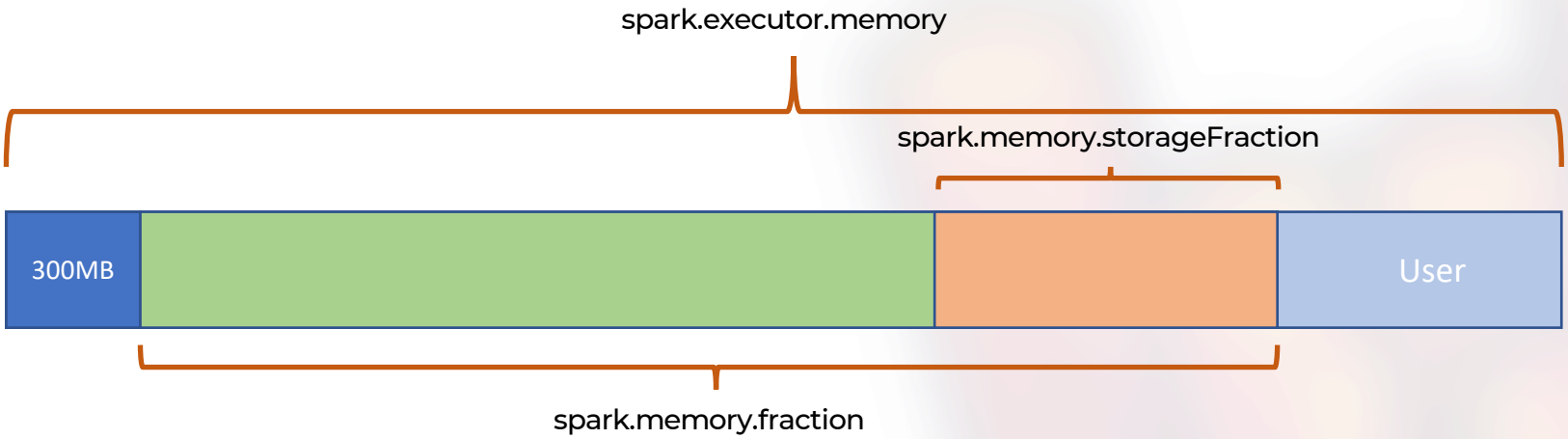
- it's possible if the compute zone is available
- a big computation would need to evict the LRU RDDs until at least the green area is free
- assume RDD1 and RDD4 are LRU



A massive computation (e.g. shuffle) must have at least the Execution area free

- if some of the Storage area is free, that's a bonus

Configurations cheatsheet



+big guns

--spark.memory.offHeap.size: for off-heap allocation (e.g. Tungsten)

--spark.yarn.executor.memoryOverhead: to request slightly more memory from the cluster manager

+driver configs

--spark.driver.memory: total allocated to driver

--spark.yarn.executor.memoryOverhead: works for driver as well

Spark rocks