|  |  |  |
| --- | --- | --- |
| Property Name | Default | Meaning |

|  |  |  |
| --- | --- | --- |
| spark.yarn.queue | default | The name of the YARN queue to which the application is submitted. |

|  |  |  |
| --- | --- | --- |
| spark.executor.instances | 2 | The number of executors for static allocation. With spark.dynamicAllocation.enabled, the initial set of executors will be at least this large.  https://community.hortonworks.com/content/supportkb/49510/how-to-enable-dynamic-resource-allocation-in-spark.html |

|  |  |  |
| --- | --- | --- |
| spark.executor.memory | 1g | Amount of memory to use per executor process, in the same format as JVM memory strings with a size unit suffix ("k", "m", "g" or "t") (e.g. 512m, 2g). |

|  |  |  |
| --- | --- | --- |
| spark.yarn.executor.memoryOverhead | executorMemory \* 0.10, with minimum of 384 | The amount of off-heap memory (in megabytes) to be allocated per executor. This is memory that accounts for things like VM overheads, interned strings, other native overheads, etc. This tends to grow with the executor size (typically 6-10%).  https://www.tutorialdocs.com/article/spark-memory-management.html |

|  |  |  |
| --- | --- | --- |
| spark.executor.cores | 1 in YARN mode, all the available cores on the worker in standalone and Mesos coarse-grained modes. | The number of cores to use on each executor. In standalone and Mesos coarse-grained modes, for more detail, see [this description](https://spark.apache.org/docs/latest/spark-standalone.html#Executors Scheduling). |

<http://www.treselle.com/blog/apache-spark-performance-tuning-degree-of-parallelism/>

|  |  |  |
| --- | --- | --- |
| spark.sql.shuffle.partitions | 200 | Configures the number of partitions to use when shuffling data for joins or aggregations. |

|  |  |  |
| --- | --- | --- |
| spark.default.parallelism | For distributed shuffle operations like reduceByKeyand join, the largest number of partitions in a parent RDD. For operations like parallelizewith no parent RDDs, it depends on the cluster manager:   * Local mode: number of cores on the local machine * Mesos fine grained mode: 8 * Others: total number of cores on all executor nodes or 2, whichever is larger | Default number of partitions in RDDs returned by transformations like join, reduceByKey, and parallelize when not set by user. |

|  |  |  |
| --- | --- | --- |
| spark.executor.heartbeatInterval | 10s | Interval between each executor's heartbeats to the driver. Heartbeats let the driver know that the executor is still alive and update it with metrics for in-progress tasks. spark.executor.heartbeatInterval should be significantly less than spark.network.timeout |

|  |  |  |
| --- | --- | --- |
| spark.kryoserializer.buffer.max | 64m | Maximum allowable size of Kryo serialization buffer, in MiB unless otherwise specified. This must be larger than any object you attempt to serialize and must be less than 2048m. Increase this if you get a "buffer limit exceeded" exception inside Kryo. |

[*Kryo serialization*](https://github.com/EsotericSoftware/kryo): Spark can also use the Kryo library (version 4) to serialize objects more quickly. Kryo is significantly faster and more compact than Java serialization (often as much as 10x), but does not support all Serializable types and requires you to register the classes you’ll use in the program in advance for best performance

To register your own custom classes with Kryo, use the registerKryoClasses method.

**val** conf **=** **new** **SparkConf**().setMaster(...).setAppName(...)

conf.registerKryoClasses(**Array**(classOf[MyClass1], classOf[MyClass2]))

**val** sc **=** **new** **SparkContext**(conf)

|  |  |  |
| --- | --- | --- |
| spark.speculation | false | If set to "true", performs speculative execution of tasks. This means if one or more tasks are running slowly in a stage, they will be re-launched. |

|  |  |  |
| --- | --- | --- |
| spark.shuffle.consolidateFiles | false | If set to "true", consolidates intermediate files created during a shuffle. Creating fewer files can improve filesystem performance for shuffles with large numbers of reduce tasks. It is recommended to set this to "true" when using ext4 or xfs filesystems. On ext3, this option might degrade performance on machines with many (>8) cores due to filesystem limitations. |

|  |  |  |
| --- | --- | --- |
| spark.eventLog.dir | file:///tmp/spark-events | Base directory in which Spark events are logged, if spark.eventLog.enabled is true. Within this base directory, Spark creates a sub-directory for each application, and logs the events specific to the application in this directory. Users may want to set this to a unified location like an HDFS directory so history files can be read by the history server. |

|  |  |  |
| --- | --- | --- |
| spark.eventLog.enabled | false | Whether to log Spark events, useful for reconstructing the Web UI after the application has finished. |

|  |  |  |
| --- | --- | --- |
| spark.driver.maxResultSize | 1g | Limit of total size of serialized results of all partitions for each Spark action (e.g. collect). Should be at least 1M, or 0 for unlimited. Jobs will be aborted if the total size is above this limit. Having a high limit may cause out-of-memory errors in driver (depends on spark.driver.memory and memory overhead of objects in JVM). Setting a proper limit can protect the driver from out-of-memory errors. |

|  |  |  |
| --- | --- | --- |
| spark.driver.memory | 1g | Amount of memory to use for the driver process, i.e. where SparkContext is initialized. (e.g. 1g, 2g).  Note: In client mode, this config must not be set through the SparkConf directly in your application, because the driver JVM has already started at that point. Instead, please set this through the --driver-memory command line option or in your default properties file. |

|  |  |  |
| --- | --- | --- |
| spark.driver.cores | 1 | Number of cores to use for the driver process, only in cluster mode. |

|  |  |  |
| --- | --- | --- |
| spark.memory.fraction | 0.75 | Fraction of (heap space - 300MB) used for execution and storage. The lower this is, the more frequently spills and cached data eviction occur. The purpose of this config is to set aside memory for internal metadata, user data structures, and imprecise size estimation in the case of sparse, unusually large records. Leaving this at the default value is recommended. For more detail, see [this description](https://spark.apache.org/docs/1.6.0/tuning.html#memory-management-overview). |
| spark.memory.storageFraction | 0.5 | Amount of storage memory immune to eviction, expressed as a fraction of the size of the region set aside by s​park.memory.fraction. The higher this is, the less working memory may be available to execution and tasks may spill to disk more often. Leaving this at the default value is recommended. For more detail, see [this description](https://spark.apache.org/docs/1.6.0/tuning.html#memory-management-overview). |