16.3.3

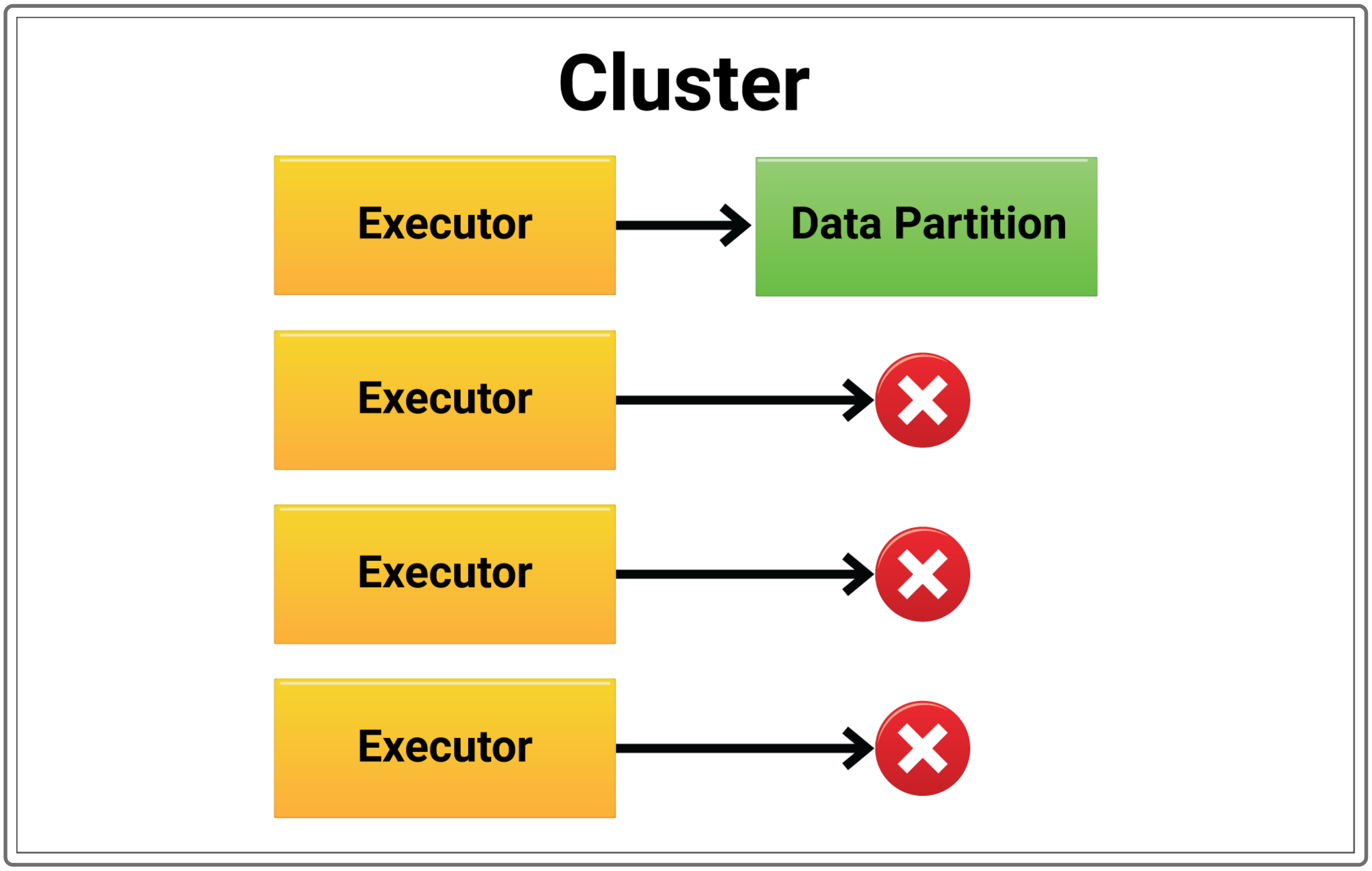
Spark Parallelism

You recall that Hadoop worked on the distributed file system and that Spark can do something similar. Continuing to learn how Spark works, you dive deeper into how Spark can work with data stored in different partitions and operate in parallel.

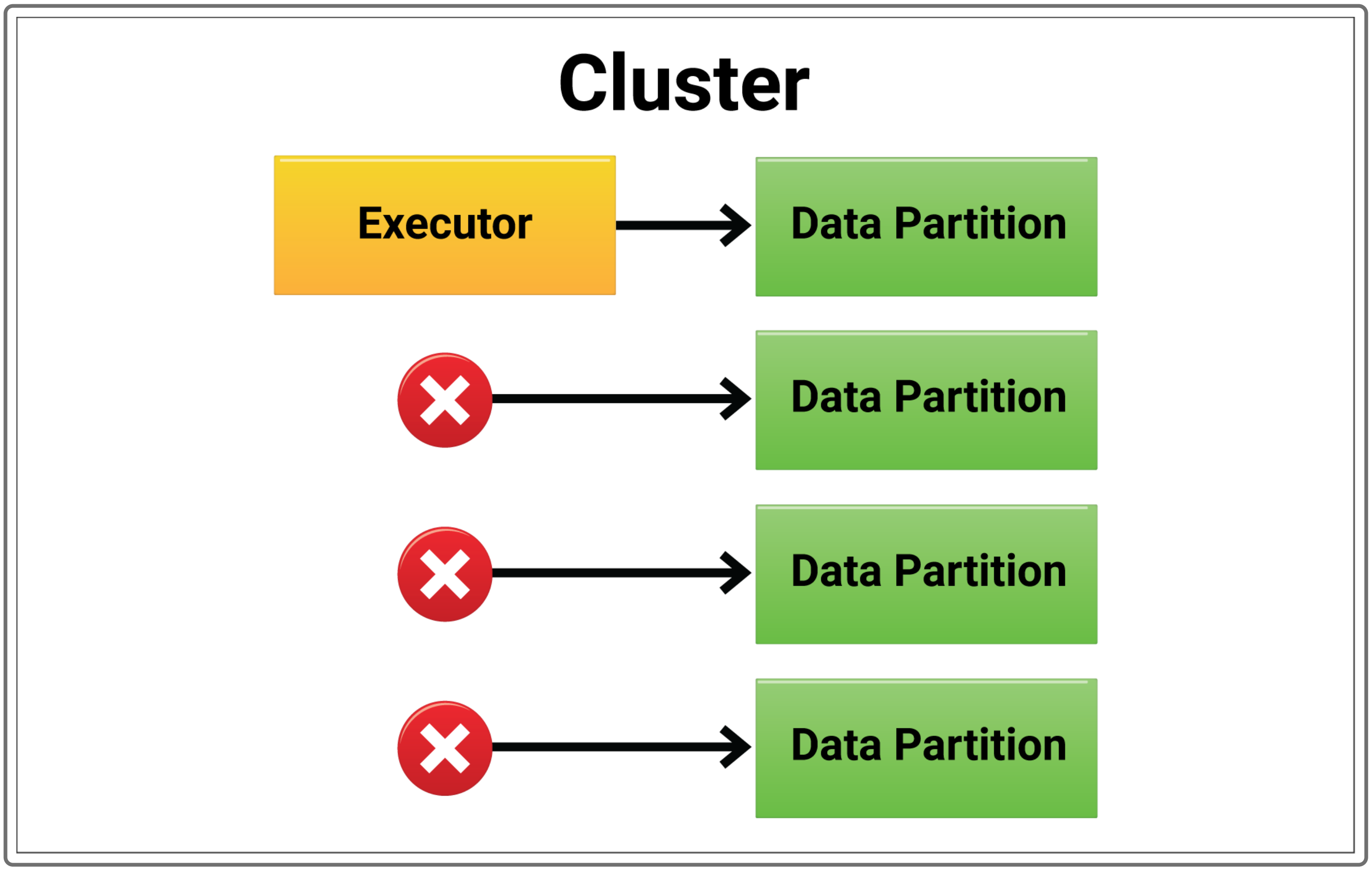
Parallelism in Spark entails running work through a cluster (group of computers) concurrently, instead of performing all work on a single computer. Recall when you and Jennifer were each counting video game reviews in two separate datasets, and you were working parallel to each other. You did not need to wait for Jennifer to do anything before you could start counting your reviews. This is exactly what Spark is doing.

Data is broken into **partitions,** meaning a chunk of your data will sit on a physical machine in your cluster. If your Spark application has three machines, each one of those machines will have a piece of your original dataset. For example, a dataset with 1,000 rows of data might have 334 rows on one machine, 333 on a second, and 333 on a third.

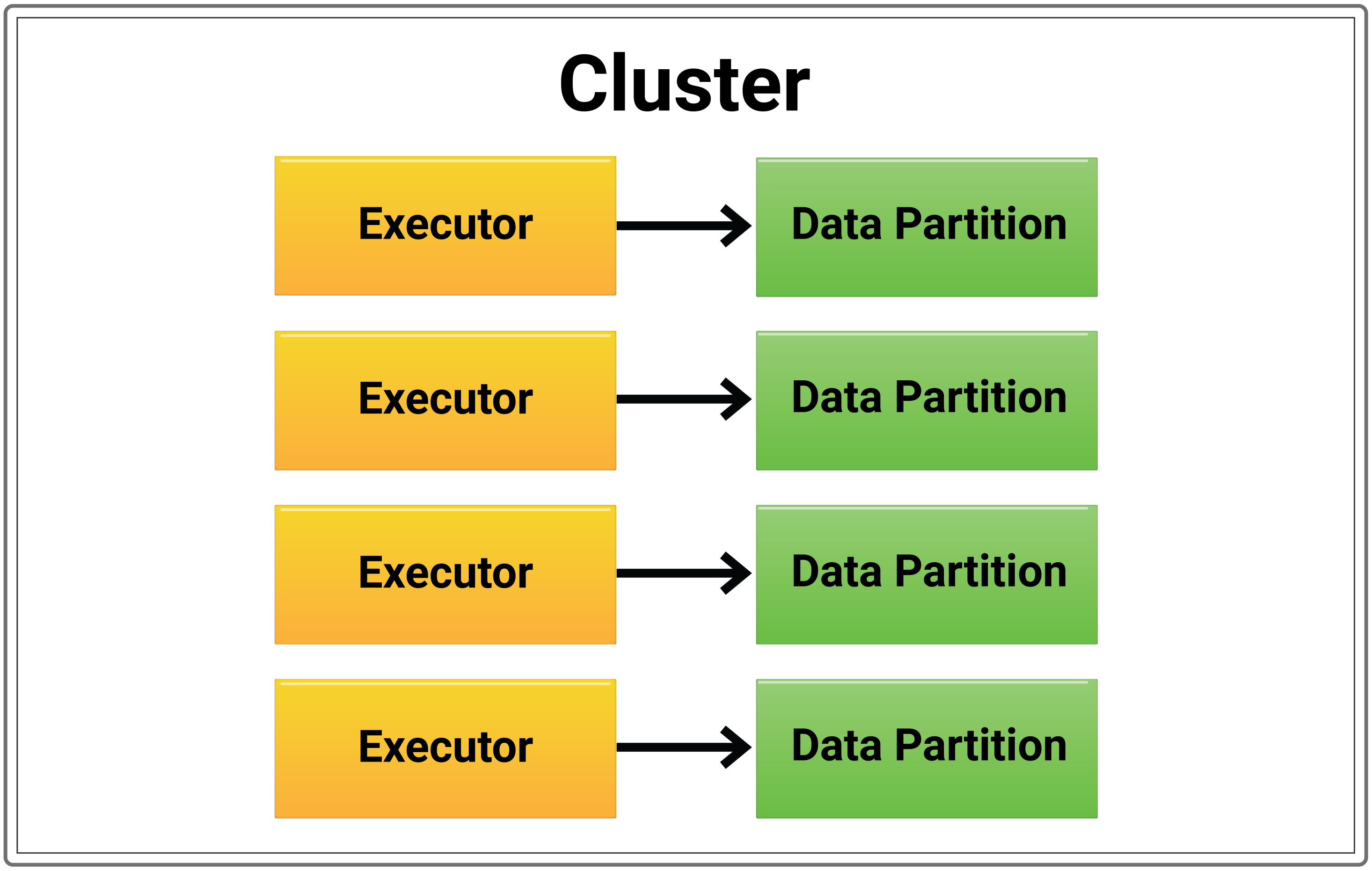
Note the following guidance when running code in parallel: If there is only one partition but many executors, the parallelism is only one. One machine can only work with one executor. The following image visualizes this concept:



If there are many partitions but only one executor, the parallelism is still only one. One executor is assigned to one machine, as shown in the following image:



Each executor needs to work on each partition for **perfect parallelism**, as shown below:



Consider the manager example: If there is only one task but many employees, that task is completed in the time frame that one employee can accomplish it in, and at the pay rate of one employee.

Next, let's take a look at Spark's API.