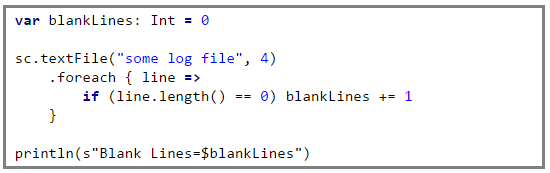
**Why is there a need for broadcast variables when working with Apache Spark?**

These are read only variables, present in-memory cache on every machine. When working with Spark, usage of broadcast variables eliminates the necessity to ship copies of a variable for every task, so data can be processed faster. Broadcast variables help in storing a lookup table inside the memory which enhances the retrieval efficiency when compared to an RDD lookup ().

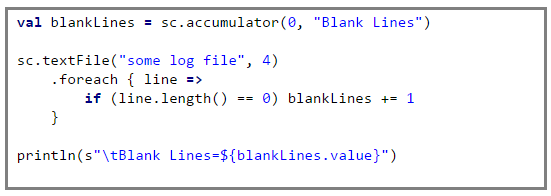
**Why use Spark Accumulators?**

Now why do we need accumulators and why not just use variables as shown in the code below.



The problem with the above code is that when the driver prints the variable *blankLines* its value will be zero. This is because when Spark ships this code to every executor the variables become local to that executor and its updated value is not relayed back to the driver. To avoid this problem we need to make *blankLines* an accumulator such that all the updates to this variable in every executor is relayed back to the driver.

So the above code should be written as,



This guarantees that the accumulator *blankLines* is updated across every executor and the updates are relayed back to the driver.

We can implement other counters for network errors or zero sales value, etc. The full source code along with the implementation of the other counters can be found [here](https://github.com/prithvirajbose/spark-dev/blob/master/src/main/scala/examples/PurchaseLogAnalysis.scala).

People familiar with Hadoop Map-Reduce will notice that Spark’s accumulators are similar to Hadoop’s Map-Reduce counters.