Q1 - Summary of how ML on TF differs from ML on Spark

Comparison among Spark and Tensor-Flow for Machine Learning models

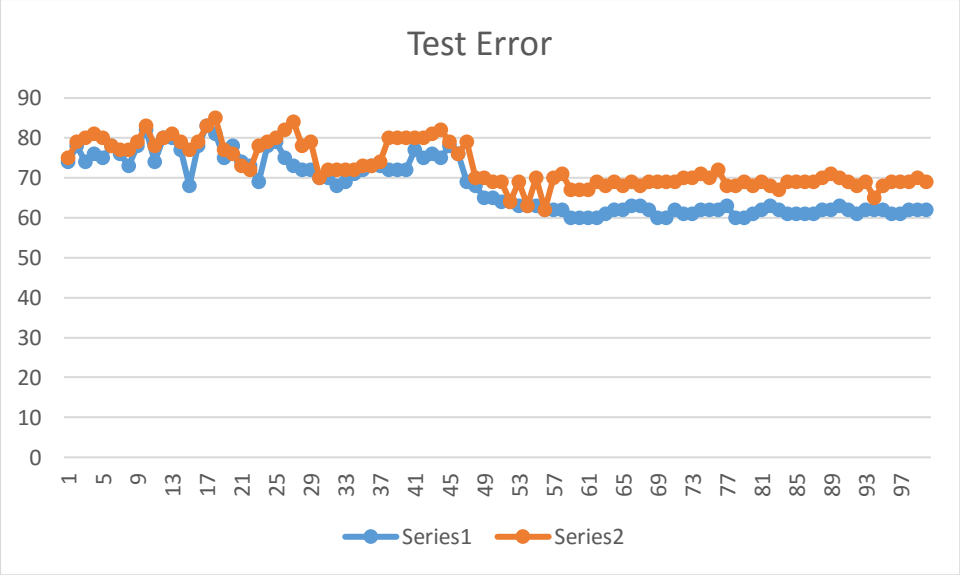
Parallelism and Synchronicity:

Spark RDDs are a good fit for many parallel applications because they naturally apply the same operation to multiple data items in memory (as seen in gradient example in spark paper).
However, Spark has only data-parallelism but Tensor-Flow has both, data and model parallelism.
Model-parallelism has benefits in case of huge models, however network communications may quickly overhead the benefits.

Spark RDDs would be less suitable for applications that make asynchronous fine-grained updates to shared state, while TF works well for both Synchronous and Asynchronous updates.
Although making the workers asynchronous helps, but if they get too out of sync it can slow or even prevent convergence.

So, to summarise, when it comes to machine learning models: Although TF has important advantages over Spark, in the form of Asynchronous Updates and Model Parallelism, but these would be undone in case of too much lack of sync and huge network communication overhead, respectively.

Q2 - Graph Plot for Synchronous and Asynch SGD for every 100th iteration



Series 1 – Synchronous SGD
Series 2 - Asynchronous SGD

Q3 - Average Values of CPU Idle time, Disk RW, NW IO in KB

	CPU Idle	Disk Read	Disk Write	NW recv	NW send
Synchronous	**95%**	**93**	**29**	**6853**	**6552**
Asynchronous	**84%**	**256**	**35**	**29316**	**28092**

From the above data, NW transfer is the bottleneck in case of synchronous and asynchronous SGD. However this bottleneck is more pronounced in case of asynchronous SGD