Ikhlas Attarwala
MSAI 339 – Data Science Seminar
10-16-18 – Background Reading

Summarizing "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing" by Zaharia et al.

The authors from UC Berkeley offered a solution for the growing demand on efficiency for computing iterative/recursive algorithms and interactive data mining tools. They offered Resilient Distributed Datasets, a fault-tolerant system for clustered computing, but the catch to making it efficient was for it to remain in-memory. Keeping intermediate data in-memory allowed them to reuse it quickly. Zaharia et al. define RDDs as read-only, partitioned records. There are 5 operations (pieces of information) to represent each RDD: partitions (number of pieces), preferred location (which nodes can access a partition fastest), dependencies on parent RDDs, iterators, and partitioners. Performance-wise, RDDs data mine through tasks 40 times faster than Hadoop, and performs on machine learning tasks about 20 times faster than Hadoop.

One strength of this paper, that I was quite appreciate for, was the breadth of schemas across the paper. Whether it be for comparisons between applications, dependency representations, or even illustration differences of in-memory partitions vs those not are not in-memory. Additionally, as the authors were rightly confident with their work, they discussed implementation of RDDs with a range of applications of various usages, including but not limited to Hadoop, MapReduce, and Pregel. There are evaluations and performance benchmarks on every one of these discussed! Finally, they wrote a great piece on related work in this field. This not only helps a novice programmer to grow familiar with the field, but also help illustrate all that RDDs can integrate with.

Unfortunately, Zaharia et al. never addressed solutions to their biggest problem. Because of the in-memory clustering, there needs to be a great deal of storage to keep it in check. They state that for all their testing procedures, they made sure each machine in the cluster had enough memory to store all RDDs across every iteration ran. In doing so, there is a selection bias. A second and related weakness is that given the issue and bias that they explicitly state, they don't make any address the issue with solutions. In the real world, there are instances where memory fails, and when it happens, there are contingencies to address the problem. Zaharia et al. do not address any such test of a machine running short on memory for this case. Finally, the paper does not discuss limitations or plans for improvement they can see in the future of RDDs. This lacking may not be great for readers who place impulsive importance on how well the RDD can improve hereafter if they aren't explicitly told.

This article was quite hard to find weaknesses in because it was well written and discuss almost all sides to RDDs. They even include example programs to help us understand how it integrates with various systems. I think they should focus on the one topic they tended to avoid (as mentioned above), possible lack of memory in machines. If they readdress this real-life concern, I think it could have a positive impact on the users this paper reaches.