**Y790 Report for 2012 Spring Semester**

**By:** Yuduo Zhou

**Report to:** Dr. Geoffrey Fox

**Topic:** Migrate Iterative MapReduce Runtime and Application onto Distributed File Systems

**Content:**

Original Twister depends solely on native file system and SCP protocol to maintain input and output data during the calculation, which could bring some advantage to iterative MapReduce like data locality and faster I/O rate. However, Twister’s dependency on native file systems also makes it vulnerable to data corruptions since no data replication is implemented. This highly limits the scalability and fault tolerance of Twister. While on the other hand, Hadoop Distributed File System is well-known for its scalability and fault tolerance. Besides, HDFS also provides more powerful file operations comparing to the Twister script we currently have. Therefore I believe migrating Twister onto HDFS will be greatly beneficial towards its scalability, fault tolerance, and functionality.

One requirement to this approach is to have minimum change to Twister, because Twister is still on-going and may change its structure in the future. Since Twister requires a partition file for scheduling, thus we developed automatically partition file generation. With respect to file distribution, the generated partition file will maximize data locality using Max-flow algorithm.

Original Twister is faster at small or medium scale. But it may fail to maintain linear scale up as we adding nodes, because it doesn’t have associated distributed file system. Hadoop is good at maintaining stable performance as it’s scaling up, but its performance is outperformed by Twister. By combining Twister with HDFS, our purpose is to maintain the good performance of original Twister while creating satisfied scalability by taking advantage of HDFS.

During this semester, I have:

* Successfully developed the HDFS package that can be attached to Twister without any change to it.
* Successfully revised Word Count, Matrix Multiplication, K-means Clustering, and MDS into HDFS compatible version.
* Implemented early stage performance test on HDFS-Twister.

Base on some typical performance data I generated, it’s observable that HDFS-Twister has very slightly performance downgrade comparing to the original one. This is understandable because HDFS I/O rate is slower than native file system on compute nodes, according to Zhenhua Guo’s work. It is a reasonable compromise if we take the advantages brought by HDFS into consideration.

Table Data Distribution Cost

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Distribution Time (second)** | | | |
| Data size (G) | 1 | 4 | 16 |
| HDFS | 20.3871 | 26.9711 | 257.374 |
| ORI | 12.8644 | 36.33 | 202.14 |

The data distribution cost records how much is needed to spread given data across 16 nodes on polar grid. Note original Twister performs this distribution using 8 threads simultaneously using SCP copy, and HDFS only has one process to do data uploading, replication and auto balancing. So naturally HDFS Twister will take more time than the original one.

The following figures represent the calculation time and overhead of k-means running across 16 compute nodes on India@FutureGrid with 1G, 4G, and 16G data.

From the above figures, we can see HDFS-Twister incurs more overhead than the original Twister, which has been explained. Also the overhead is stable, which will not impact the overall performance too much if the scale goes up.