Poodah Map Reduce Framework

Raj Agarwal (rajagarw), Kirk Zhang (keweiz)

Our poodah distributed framework meets all the requirements listed in the handout. As long as the master does not crash, our system will continue to work. If any errors occur, error messages will be sent to the process manager. Our system will still try to complete the job, but the result might not be fully complete.

However, there are some limitations with our design. One limitation is that we do not give the application user the ability to perform local combining during the map step. Another limitation is that we can only accept one file at a time, not a whole directory. Once a job is received by the master, the master will parse the file and send them to different mappers. When a mapper has completed its task, it will send the result back to the master for sorting. The master cannot begin sorting until it gets all the results back. Likewise, the master cannot produce the final output file until it gets all the results back from the reducers. Our master only has one shuffle listener that performs the job sorting. Thus, only one job can be sorted at a time.

Our framework also uses fixed size record files, which is set by the job configuration. Whenever a key value pair is generated, we have be able to serialize the key value to an object and write it to a file. To ensure uniformity, we always allocate a fixed size byte array for each key value object. A problem can occur when the object's size exceeds the fixed size byte array. When this happens, an error message will be sent from the participant to the master which is redirected to the process manager. Additionally, using this fixed size record file method approach requires a lot of disk space. It is possible that your system crashes because you reach your AFS disk quota limit.

Finally, our system does not handle the case when the master crashes. If we had more time, we could have developed a fault tolerant system that elects a new master when the master crashes.