Machine Problem 4 Report

1. Design

In this MP4 we design a distributed Crane system in order to conduct distributed application. We design 3 different kinds of application that contains filter, transform and join function. We also install Spark on our machines and compare it with our Crane system with the same topology and application. In our design, we adopt the previous system structure in MP1-MP3, MP1 for debug, MP2 for construction failure detection system, MP3 for distributed file System.

1) Overall Structure

Our Crane System have 3 major constructions, failure detection, file distributed, application topology pipeline. The former 2 parts has been realized in previous MP and adopted into Crane. We used Storm-liked structure to build Crane System. Our System contains one master, one standby master, and workers. Master assigns task to workers and standby master would replace master when master is down. Standby master would have a copy of all information in master so that it can replace the master immediately when master is down. For each application, our system would allow client to enter the topology of realization, which is to allow user to define the content and number of spout and bolt. When client finishes defining the spouts and bolts for topology construction, and submit the job in our Crane System, master in the system would receive the job and distributed task to each worker, assigning them as spout or bolt. Each worker would know its job and the next worker that it needs to send the processed data to. The number of bolts in each procedure would be distributed by master as well. The stream data comes from our distributed file system, and the result from application would be put into the distributed file system as well.

2) Failure Detection

When application is running in our Crane System, if one or more worker fail, the master would know through the failure detection system. And master would construct a new topology and send new task to workers to complete the original application. When workers received the new task assignment, they would stop the old task and start working on the new task. When master is down, standby master would replace the position of master, and the original application would go on without interruption.

3) Filter Application

We design a filter application to select the content that we want to maintain in the stream data. This application is very common in real life. It can be applied to select important articles in a big database, search keyword to find content through Google and so on. In the application,

user need to input the word he or she want to mine. Our application would keep the sentences that contain the word and ignore the other sentences in the stream. Master would assign multiple workers as bolts to deal with the data stream simultaneously, and then use a worker to combine the filter result and put the result into our distributed file system.

4) **Word Count Application**

This word count application is for counting the occurrence of word in the data stream. Our System would assign one spout to input the stream, multiple bolts to map the word into key-value pairs, and one bolt to combine the pairs and get the final counts, and put the result file into our distributed file system.

5) **Join Application**

This join application is for joining data streaming and content from the local file in our system. This application is common in application interacting database in the real world. Our System would assign one spout to input the stream file, multiple bolts to join every line that comes from spout with every line in the local file. Every bolt has the same local file. And we also use one bolt to combine all the result together and write the result into the distributed file system.

2. Experiment

We compare the performance of 3 applications in our Crane system and Spark Streaming, we use 1MB, 5MB, 10MB, 20MB, 50MB files to test the 3 application running time on Crane and Spark Streaming. We apply the same topology on Crane and Spark Streaming. We use 5 machines in a cluster to get the following result.

1) Filter Application

Time(second)	1MB	5MB	10MB	20MB	50MB
Crane	1	3	3	6	5
Spark	21	14	18	15	18

From the result we can see that Spark Streaming have very stable running time with 1MB to 50MB files. But Our Crane System is faster than Spark Streaming when using files between 1MB and 50MB.

2) Word Count Application

Time(second)	1MB	5MB	10MB	20MB	50MB
Crane	2	4	7	16	68
Spark	20	24	26	24	31

We can see that Spark Streaming have very stable running time with 1MB to 50MB files. Our Crane System is faster than Spark Streaming when using files between 1MB and 20MB. But our System is much slower than Spark when the file size is 50MB.

3) Join Application

Time(second)	1MB	5MB	10MB	20MB	50MB
Crane	1	2	4	12	60
Spark	23	17	16	20	17

From the result we can see that Spark Streaming also have very stable running time with 1MB to 50MB files. Our Crane System is faster than Spark Streaming when using files between 1MB and 20MB. But our System is much slower than Spark when the file size is 50MB.

4) Comparison Among 3 Applications

From the following plot we can see that Spark performs very stable in 3 applications. But in our Crane System, filter application performs better than the other two applications. The reason may be that filter application selects and reduces the data when processing stream.