Greenlee 560 Summary 5

Elliot Greenlee

February 27, 2017

1 GraphX

2 Problems

What problems does this paper try to solve? Why are such problems important?

This paper attempts to address the needs to the entire analytics pipeline. Previously, the increases in scale and importance of graph data have led developers down the path of specializing graph processing systems. This has left out the rest of the analytic process, meaning that pipelines often have unnecessary data movement or duplication. Also, fault tolerance is generally abandoned, and the support of distributed dataflow frameworks is lost. Each of these losses were considered acceptable with the performance gains, but GraphX attempts to recover the losses.

3 Assumptions

What are assumptions made by this paper? Are they verifiable? Are there logical holes in such assumptions?

The papers main argument is that many of the advantages of specialized graph processing systems can be recovered in a modern general-purpose distributed dataflow system. The full description and results of implementing a system to test this exact argument show that their assumption was correct. Through experimentation the paper shows that graphX can match the performance of specialized graph systems.

4 Solutions

What are the major solutions of this paper? Do you think the solutions in this paper will work for the problem? Do you think the paper evaluates the solutions in a convincing manner? List two limitations of the solutions proposed in this paper, and outline your method to fix them.

The paper describes GraphX, which allows users to model data to fit the entire analytic pipeline by choosing the pattern that is best suited for the current task. They unify advances in graph processing with advances in dataflow systems.

4.1 Limitations

Currently GraphX will only work on graphs that are past the creation process. Adding functionality for streaming seems like a key way to further develop the software and generalize to even more common patterns.

GraphX only runs tasks synchronously; other programs allow asynchronous execution to address stragglers. While GraphX handles these stragglers with other functionality later, asynchronous execution would allow slow processes to be handled by redistributing after the initial call.