Reading Summary of GraphX: Graph Processing in a Distributed Dataflow Framework

T11 PangCloud: Tianyang Liu, Su Pu, Yicheng Wang

link: https://docs.google.com/document/d/1xXa-shLQFdCzt0qhqYz0AtOGej-5OsCufPDwIKStctE/edit

Spark GraphX is a distributed graph processing framework. It is based on the concise but very rich interface provided by Spark for graph computation and graph mining, and has greatly simplified need toward distributed graph processing. It is well acknowledged that in a social network there could have lots of relationship links between people, like Twitter, Facebook, Weibo, and Wechat. These places where big data generate have great demand on graph computation. Nowadays graph processing is mostly distributed rather than on a single node. Spark GraphX uses Spark as the basis, therefore it is a distributed graph system naturally. Distributed storage of a graph adopts point segmentation method and uses partition By method which allows users to select various partition strategies. This strategy will allocate edges to its edge partition, vertex to its vertex partition. Partition of the edge also caches local ghost copy of edge linking point. Different strategy of segmentation will influence the number of cached ghost copies in demand, and the balance degree of edges in each edge partition. So it is good to select the best solution according to the specific structure of a graph. Currently we have four optimized strategies including: EdgePartition2d, EdgePartition1d, RandomVertexCut, and the CanonicalRandomVertexCut.

The distributed processing and parallel processing of a graph is in fact to segment a graph into numerous small ones, then we do computation on each of them respectively. In each computation we may do iterative periodic computation. In the example from the paper, we can see that, after we get Wikipedia documents, we change it into link tables form of view, then upon which analyze into hyperlink, finally we should use PageRank to get top communities. The route process below from editor graph to community, is called triangle computation which is an algorithm to calculate triangles, based on which we can find communities. From what we have discussed we see that there are multiple ways and algorithms to do graph computation, meanwhile, how to do transformation between graph and tables.

In the design of GraphX, both point segmentation and GAS are fairly mature. Designers optimized them in the design and coding, and found the best balance between function and performance. Just like Spark itself, each sub-module has an abstraction. The core abstraction of GraphX is called Resilient Distributed Property Graph, a directed multilayered graph where each point and edge has attributes. It expands the abstraction of Spark RDD and has two views of table and graph, but only one copy of physical storage. Two kind of views have respective special operators, therefore gain flexible operations and execution speed. Graph also has very clean codes. The core code of it is just 3k lines approximately, and the Pregel mode based on this has as few as 20 lines. Most of whose realization is surrounding partition optimization. It to some extend proves the point segmented storage, and the corresponding computation optimizations are the most difficult part of a graph processing framework.

There are several key points in the base layer design of GraphX. All operations on graph view would eventually be translated into its related table view’s RDD operations. Therefore, the computation of a graph in its logic can be compared to series of RDD transformations. As a result, graph owns three key features of RDD: immutable, distributed, and fault tolerant, immutable as the most crucial. Logically, all transformations and operations will generate a new graph; Physically, GraphX to some extent has optimization of invariant vertex and edge, both would be transparent to users. The physical data shared by the basis of two views are composed by the RDD vertex partition and RDD edge partition. Point and edge are intrinsically not stored as collection, but a piece of data block with indexed structure in the internal storage, to accelarate the iteration speed under different view. Identical indexing structures are shared in RDD transformation, therefore decreases the cost of computation as well as the storage.

Three strong points of this paper include:

1. Deal with old traditional graph processing problems

The general graph processing is that it cannot scale well and do not support for the fault tolerance. The distrbuted dataflow framworks can face this problem by providing tailored programming abstractions and accelerated the execution of iterative graph algorithms. The paper realized to build the GraphX on Apach Spark, the GraphX gain some advantages from the Spark, it can let people modify and see the computational pipleline.

2. The benefit of GraphX

The GraphX contains the properties such as vertex collection and edge collection, these normailized representation of property graph as a pair of vertex and edge property collection allow user to embed grahs in a distributed dataflow framwork. GraphX can also express graph-parrallel compution in a distributed dataflow

framework as a sequence of join starges and group-by stages punctuated by map operations. It can natually easy to compute the PageRank Twitter follower graph and other structures.
3. The GraphX achieves fault tolerance.

Since GraphX are built on Spark, which provides lineage-based tolerance with negligible overhead as well as optioanl dataset replication. The GraphX also achieve this very important attribute compare to other graph processing systems.

Three weak points of this paper include:
1. Since the GraphX is designed for general graph processing, it may take more time on running compared with specialized graph processing system.
2. GrahpX focus on parallelizing the computation by partitioning edges among machines, which would leads to a high cost on communication
3. The GraphX has problems on memory consumption even with lo-degree vertices.

References:
[1] http://blog.csdn.net/tanglizhe1105/article/details/50740295
[2] http://lib.csdn.net/article/spark/3082
[3] Chen, Rong, et al. "Powerlyra: Differentiated graph computation and partitioning on skewed graphs." Proceedings of the Tenth European Conference on Computer Systems. ACM, 2015.