Finalized Proposal

*Team members: Linna Hu. Hongyu Huang. Jiashuo Sun.*

1. **What is the problem?**

Increasing number of data content and user volume makes it necessary to envelope an optimized and more efficient distributed computation system. Nowadays, machine learning plays an important role in intelligent service. Improved Topic-modeling and click-through prediction, a family of algorithms, make it more convenient for users to enjoy the smart service. But since big data is growing up quickly, traditional algorithms could no longer handle such thing.

Graph computation system, as one of branch of distributed system, its scale of settlement is increasing gradually. Although traditional graph applications, like PageRank, could handle graph computation efficiently, applications of machine learning can’t use them directly because of its different computation models, such as mini-batch and SSP.

This paper is about , which is a new distributed graph engine that bridges graph computation and distributed machine learning.

**2. Why is it important to solve?**

Large data volume leads to large load. If we don’t put data into new algorithm models, it will take much more time to compute them and will even result in a huge number of servers striking. is designed to preserve the benefits of graph engines while extending their data models, programming models, and scheduling approaches in service to distributed machine learning.

For Heterogeneous data layout, supports heterogeneity in multiple dimensions of data layout, instead of just assuming a homogeneous graph. Getting rid of complicated method of enumerating edges, uses a bipartite graph to just scan vertices. This performance makes a big progress in data layout.

To deal with the inflexibility of addressing deficiencies of GAS problem, introduces MEGA model, which avoids complicated phases procedure and non-scatter phases problem. It allows an arbitrary sequence of stages.

Compared with BSP, using SSP to execute each iteration on a mini-batch with specified size to manage to avoid mess-updated problem and late iteration operation.

Three algorithms for intelligent service are introduced in . They have solved the problem of traditional algorithms not able to compute with machine learning model, for its difference from traditional computation models.

In sum, is a critical step on the region that converge the graph computation and distributed machine learning. This creative optimization gives us a new chance to deal with big data and compute graph more efficiently.

Even though we want to check if the performance between running dataset using one computer and a few of machines is different, ***Tux^2* is not a open source**. So we switch our goal to GraphLab, which is also a flow processing parallel computation framework for machine learning.

**3. Any initial thoughts on what you want to do?**

① Deploy GraphX.

② Implement some interesting applications on GraphX to understand the graph handling procedure.

③ Compare the properties of GraphX with TuX2, pick up some of them that TuX2 has not involved.

④ Raise up some suggestions of GraphX from the experiments we did.

⑤ List out the methods to optimize GraphX, where TuX2 has not referred to.

**4. How would you evaluate your solution？**

① Success to deploy Spark and GraphX.

② Run the application successfully.

③ Notice something should improve but TuX2 has not involved, and give some ideas of implement.