The past decades have witnessed the massive growth of Internet. Vast amount of graph data was produced by the boom of social network, e-commerce and online education. To process these data, there are two branches of development. One focus on computation like PageRank or other BSP algorithms on the whole graph. Distributed graph computing systems like Pregel and GraphLab are proposed for this branch. The other branch focus on management of graph data, providing support like OLTP and XXX query. In this branch, graph databases like Neo4j and Titan are developed for management of property graph.

The property graph is a directed, labeled graph with multi-edges, i.e., edges with the same source vertex and destination vertex. Vertices and edges can be associated with any number of properties. Since it can represent graph data in most scenarios, the property graph model has numerous applications in industry.

However, traditional graph databases encounter significant performance degradation when the graph contains large amount of multi-edges.

We reveal the cause of this in Titan, an open source distributed graph database, which has attracted wide attention in industry. Furthermore, we propose HybriG, a better distributed architecture based on Titan and HBase for this scenario.

Titan stores graphs in adjacency list format, where a graph is stored as a collection of vertices with their adjacency lists. Each entry of the adjacency list stores an edge or a vertex property.

When querying about adjacent vertices, Titan has to look through the entire adjacency list of the source vertex, (this cost can not be ignored when multi-edge set grows explosively) which hurts the performance when the multi-edge set grows explosively.

(separate graph data into three part: vertex data and edge data, graph structure?)

HybriG implements property graph model as well base on Titan and HBase, one of the storage engine of Titan. HybriG stores the vertex data and graph structure in Titan, the edge data in HBase, respectively.

This helps HybriG to keep a concise adjacency list about the graph structure, which helps to gain an order of magnitude improvement in execution of adjacent vertices based queries and batch loading of edge set. The scarify of this separation is that we should guarantee the data consistence between Titan and HBase.

In this article, we introduce how HybriG implements its storage layer upon Titan and HBase, and how HybriG transforms graph operations into Titan and HBase APIs. (duplicated)

Besides, HybriG achieves high performance for inserting batch of graph data.

Furthermore, (introduce consistency problem). we introduce the implementation of data consistency between Titan and HBase.

Extensive experiments have been conducted to show the outstanding performance of HybriG.