Graph Partitioner and Distributed Traversal for Distributed Graph Database Array

Dheeraj Kumar Cidda
Texas Tech University
Department of Computer Science
Lubbock,TX, 79415
Email : dcidda@ttu.edu

Abstract— **Extensive Graph related applications such as Complex networks,Biological network, social network, PPI network etc, Web Page Ranking, VLSI Design require distributed database arrays for storing the huge graph they need. One common issue with the graphs is that how they are partitioned such that distributed graphs database can be efficiently used at its full extent. In this topic, we implement a middle ware called graph partitioner. This middle ware will consists of many graph partitioning algorithms by which the graph can be partitioned into clusters and stored in the distributed database array. This process helps the end users in accessing the graph data which are partitioned according to different partitioning strategies through some specific procedures. Further, we also evaluate the performance of the distributed graph database array against different graph partitioning algorithms.**

I. Introduction

Graph partitioning is an essential preprocessing step for distributed graph computations. The cost of fine-grained remote memory references is extremely high in case of distributed memory systems, and so one usually restructures both the graph layout and the algorithm in order to mitigate or avoid inter-node communication.

A. Proposal

In this topic we implement a graph partitioner using some graph partitioning techniques. Further, after diving the graph into clusters we distribute these clusters along various distributed systems and at the end we evaluate the performance of the algorithms used and also the efficiency of distributed systems.

B. Deliverables

Week 1 : Study about the Graph Partitioning Problem and also implementation in distributed systems and start implementing it.
Week 2 : Designing a machine for BFS traversal.
Week 3 : Traversal operations after the graph partitioner algorithm implementation
Week 4 : Do the performance evaluation for the algorithms, check the efficiency of the distributed systems and do the final documentation.

The Basic definition of a graph partitioning problem is that if we consider a graph G = (V, E), where V denotes the set of n vertices and E the set of edges. For a (k,v) balanced partition problem, the objective is to partition G into k components of at most size v (n/k), while minimizing the capacity of the edges between separate components. Also, given G and an integer k ¿ 1, partition V into k parts (subsets) V1, V2, ..., Vk such that the parts are disjoint and have equal size, and the number of edges with endpoints in different parts is minimized. In other words,we can say that dividing the work evenly and minimize communication between the nodes. Partition into two parts is called graph bisection, which recursively applied can be turned into algorithms for complete graph partitioning.

Inter-node communication time constitutes a significant fraction of the execution time of graph algorithms on distributed-memory systems. Global computations on large-scale sparse graphs with skewed degree distributions are particularly challenging to optimize for, as prior work shows that it is difficult to obtain balanced partitions with low edge cuts for these graph. We use Breadth-First Search (BFS) as a representative example, and derive upper bounds on the communication costs incurred with a two-dimensional partitioning of the graph.

II. Background

In terms of the object on which the graph cut is carried out, basically, the graph partitioning algorithms can be divided into two categories, and one of them is called edgecut, with the other called vertex-cut.

The k-way edge-cut partitioning problem is defined as follows: Given a graph G = (V,E) with $\mid V \mid$ = n, partition V into k subsets, V1, V2,...,Vk such that $Vi \cap Vj$ = $\emptyset$ for i $\neq$ j, $\mid Vi \mid$ = n/k, and $\cup$i Vi = V , and the number of edges of E whose incident vertices belong to different subsets is minimized.

Similarly, the k-way vertex-cut partitioning problem is defined as follows: Given a graph G = (V,E) with $\mid E \mid$ = n, partition E into k subsets, E1,E2....,Ek such that $Ei \cap Ej$ = $\emptyset$ for i $\neq$ j, $\mid Ei \mid$ = n/k, and $\cup$i Ei = E, and and the number of vertices whose incident edges belong to different subsets is minimized.

There are four different graph partitioning algorithms which are divided by two categories: edge-cut and vertex-cut. They are as follows: 1. Multilevel Partitioning Scheme 2. Balanced Load Propagation 3. Streaming Graph Partition and finally Structural Balanced Vertex-cut based Graph Partitioning.

III. Conclusion

IV. References