Big Data Processing (IE494)

Project Proposal

Divyesh Ramani (202201241)
Manan Patel (202201310)

22 September 2024

Problem Area

- The traditional graph processing algorithms face lots of shortcomings, especially when dealing with large-scale graphs such as those used in Sensor Networks, Social Networks, Bio-Chemical Networks, etc. Primary problem is that of scalability. As graphs grow larger, fitting them in the memory of a standalone system becomes a challenge. Apart from the memory limitation the increasing computational cost would be too much a single system would be able to bare.
- This project's primary focus in on implementing a distributed graph processing algorithm using Spark. Utilising the parallel processing and in-memory computation properties of Spark, we can overcome the issues of scalability.

Group Members

1. Divyesh Ramani - 202201241
2. Manan Patel - 202201310

Expected Outcomes

Developing a distributed graph processing algorithm using Spark is expected to deliver several key outcomes:

1. Improved Scalability
    - The algorithm should handle massive graphs (e.g., social networks or web-scale graphs) that exceed the memory and computational capacity of a single machine by distributing the graph across multiple nodes in a cluster.

2. Efficient Use of Resources
   - Optimal use of memory and CPU across the cluster through task distribution and in-memory data storage, minimizing the need for disk I/O and enhancing computational efficiency.
3. Fault Tolerance
   - RDD structure in Spark is known to be fault tolerant, which ensures that if a node fails, computations can be recovered without restarting the entire process. This ensures reliability in large-scale graph processing tasks.
4. Data Locality and Reduced Network Overhead
   - By processing graph data close to where it is stored, Spark minimizes data shuffling and network communication between nodes, improving overall efficiency and reducing bottlenecks.
5. Support for Dynamic Graphs
   - The ability to handle dynamic or evolving graphs where nodes and edges may be added or removed frequently, without requiring full recomputation, which is especially useful in real-time applications.

These outcomes ultimately lead to better scalability, reliability, performance, and developer productivity in large-scale graph processing applications.

Selected Reading

Balaji, Janani, and Rajshekhar Sunderraman. "Distributed graph path queries using spark." 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC). Vol. 2. IEEE, 2016.