The emerging applications for large graphs in big data science and social networks has led to the development of numerous parallel or distributed graph processing applications. The need for better visualization of graphs has driven the need for such systems to be more parallel and more scalable. FemtoGraph is a framework for graph processing that is based on the Pregel Model of graph processing and can scale to large core counts and provide a framework for later incorporating distributed processing on multiple nodes.

### Related Work

- **GraphLab**: Single node or parallel asynchronous vertex-centric framework
- **Apache Giraph**: Application on top of Hadoop for dedicated graph processing
- **Apache Spark**: MapReduce based parallel framework using hdfs filesystem. While not designed for graph processing alone, a graph processing algorithm can be thought of as a series of chained MapReduce functions
- **Pregel**: A type of vertex-centric model

### Initial Problems

- Initial Fans_graph could not scale past 4 cores
- Very bad scaling
- Initial message queue used global mutexes (not lockfree)
- Little improvement
- Global mutex is not fast enough for 8 threads.

### Evaluation

- Message queue has been confirmed to be the bottleneck
- Eventually settled on Boost lockfree queue
- Local mutexes for areas of message queue too complex to implement
- Highlighted area shows message queue blocking until mutex is freed

### Future Work

- Distributed shared memory no bad topic to try
- Use parallel structures like distributed hash tables for graph storage
- We could manage some of the load imbalance by asynchronous

### Vertex-Centric Algorithms

- **Vertex-Centric Algorithms**: think like a vertex
- **Vertices** and **Edges**, each with a vertex or edge compute function
- **Vertex-Centric Algorithm**: involves each vertex independently running an algorithm
  - Each vertex can locally analyze the graph
  - All compute functions must be called before next superstep

### Pregel Model

- All compute functions must be called before next superstep
- **Compute function** is a user defined function run in the context of each vertex
- **MapReduce** is step by step parallel computation
  - In Pregel, a graph is partitioned into supersteps
  - Each superstep is broken down into a series of vertex and edge compute functions
  - At each superstep, vertices access previous state
  - The compute function is run in the context of each vertex
  - **Vertices** can send messages to vertices in the next superstep
  - **Edges** are not allowed to return state to the compute function

- **Vertices** can **un-halt** upon receiving a message
- **Vertices** can be **halted** on halts receiving a message
- **Vertices** can be **halted** on halts receiving a message

### Implementation

- **GraphLab** is based off of the Pregel Model of graph processing
- **Vertex-Centric Algorithms**: think like a vertex
- **Vertices** and **Edges**, each with a vertex or edge compute function
- **Vertex-Centric Algorithm**: involves each vertex independently running an algorithm
  - Each vertex can locally analyze the graph
  - All compute functions must be called before next superstep

### Testing Conditions

- **Testing was done with the Pagerank algorithm as a reference algorithm.**