FemtoGraph:
A Pregel Based Shared-memory Graph Processing Library

Alex Ballmer, **Ioan Raicu**, **Benjamin Walters**
Department of Computer Science, Illinois Institute of Technology, Chicago, IL

Abstract

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 faster manipulation of graphs has driven the need to scale across large core counts and many parallel machines. While distributed memory parallel systems continue to be used for high performance computing, some smaller systems make use of shared memory (SMP) and larger core counts. We have implemented a graph processing framework for shared memory systems capable of scaling past 48 parallel cores. This system leverages and scale to large core counts and provide a framework for later incorporating distributed processing across multiple nodes.

Vertex-Centric Algorithms

FemtoGraph is based off of the Pregel Model of graph processing

Vertex-centric algorithms: *think like a vertex*

- Vertex-centric algorithms run within the context of a single vertex or groups of vertices
- Computation occurs in steps
- *Synchronous* algorithms mandate each vertex must finish computation before executing the next step
- *Asynchronous* algorithms allow vertices to finish computation before the next step

Pregel Model

- A type of vertex-centric model
- Computation occurs in synchronous ***supersteps***
- ***Compute function*** called in parallel in the context of each vertex
- All compute functions must be called before next superstep
- Vertices can update edges and neighboring vertices
- Vertices can send messages to vertices in the *next* superstep only
- Vertices can ***vote to halt***. Vertices un-halt upon receiving a message
- When all vertices are halted, the simulation ends

Pagerank

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

- Pagerank is used in many real-wold machine learning and data analysis situations and algorithms.
- Pagerank ranks vertices in a graph in order of relative connectedness
- Defined by $\forall t \in P : r^{(0)}(t) = (1-\alpha) \cdot r(t) + \alpha \sum_{(s,t) \in L} \frac{r^{(i-1)}(s)}{|L(s)|}$

Related Work

Other Graph Processing Systems

The field of graph processing is very well defined with many existing commercial and opensource applications and tools available.

- **GraphLab** Single node or parallel asynchronous vertex-centric framework
- **Hadoop** 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
- **Apache Spark** MapReduce based parallel framework faster than Hadoop with in-memory processing
- **Apache Giraph** Application on top of Hadoop for dedicated graph processing

Comparing Application Performance

We chose to use GraphLab as a reference point to judge application performance as it is designed to run in a shared memory configuration on a single node.

Implementation

- Written in C++ with Boost libraries
- Graph stored in adjacency list
- Message queue based on multiple Boost lockfree queues
- Uses normal std::thread for multithreading.
- Compute function is a user defined function run in the context of each vertex
- Compute functions are called in parallel

Initial Problems

Initial FemtoGraph could not scale past 4 cores

- Very bad scaling
- Tried various graph storage techniques and threading patterns
- Little improvement
- Initial message queue used global mutexes (not lockfree)
- Profiling using callgrind showed bottleneck in message queue

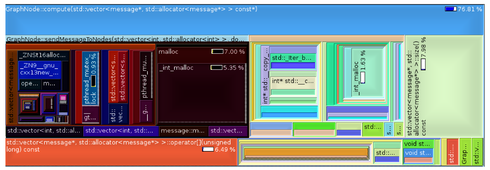

Fig. 99: Callgrind profiling

- Highlighted area shows message queue blocking until mutex is freed

Testing Conditions

All trials were done with

- The Pagerank algorithm with epsilon 0.5 and damping factor 0.5
- The Wikipedia Talk Network dataset from Stanford SNAP collection
- A 2 socket 48 core AMD server with NUMA

Future Work

- Distributed shared memory is a hot topic in HPC
- Combines SMP with a distributed memory paradigm like MPI.
- FemtoGraph can be converted to distributed shared memory
  - Use parallel structures like distributed hash tables for graph storage
  - use parallel message queue like RabbitMQ for messagequeue

Evaluation

- Message queue has been confirmed to be the bottleneck
- Global mutex is not fast enough for > 8 threads.
- Local mutexes for areas of message queue too complex to implement
- Eventually settled on Boost lockfree queue
- Performance improved immediately
- Can scale to 48 cores

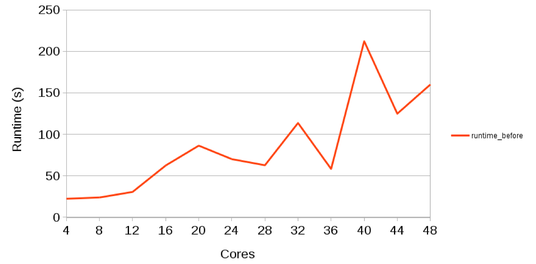

Fig. 100: Before lockfree queue

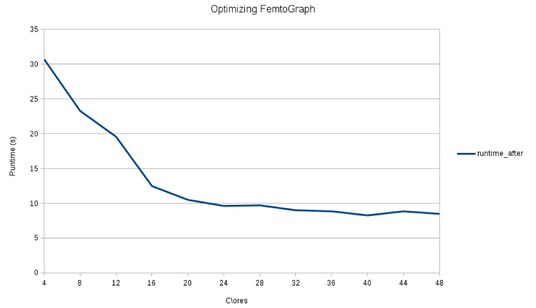

Fig. 101: After lockfree queue