High Performance Processing of Streaming Data

Supun Kamburugamuve, Milinda Pathirage, Saliya Ekanayake and Geoffrey C. Fox
School of Informatics and Computing, Indiana University

I. Overview

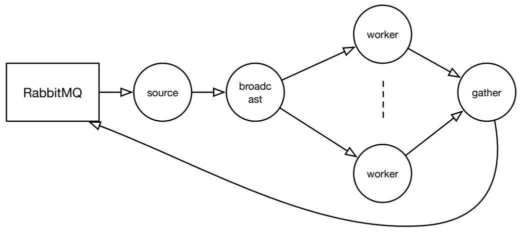
With recent advancements in cloud-based technologies and the Internet of Things, there is an emergent class of applications where real-time computing requirements of mobile robots and other IoT related devices are offloaded to clouds in order to improve mobility of the devices and accuracy of computing tasks. Due to the event-based nature of these applications, modern distributed stream processing frameworks (DSPF) such as Apache Storm provide an effective platform to meet their needs. A good example is a parallel real time distributed streaming algorithm implementing simultaneous localization and mapping (SLAM) for mobile robots [1] in the cloud. This algorithm requires low latency parallel processing with strict guarantees. Current DSPFs are designed to cater to traditional event processing tasks, such as extract transformation and load (ETL) pipelines, counting, cardinality estimation, frequent itemset finding, windowed aggregations and joins or pattern detections. The above-mentioned novel applications with strict real time guarantees demand support for low latency synchronous and asynchronous parallel processing of events, requiring efficient broadcast and gathering of data streams. Our research into high performance computing on streaming data has found that the lack of efficient communication for data distribution operations like broadcasting in current DSPF implementations is limiting the performance of these applications when the parallelism of the processing increases.

Distributed streaming applications are generally defined as a directed graph where process elements are connected through edges. Data flow occurs through the edges of the directed graph as streams of events. With naive implementations, a collective operation such as broadcast happens through separate communication links (edges) from the source to each target. Such communications can be made efficient by modeling them based on data structures such as trees. In the last couple of decades, different broadcasting algorithms have been perfected for HPC applications using technologies like MPI. In this work we propose to remodel these algorithms to improve broadcasting for DSPFs considering attributes of the special class of streaming algorithms mentioned above and those of cloud runtime environments.

We implemented several broadcasting algorithms for an open source fork of a DSPF called Apache Storm. The primary algorithms we implement are flat tree with node-based variation, binary tree and pipeline. Storm utilizes both process based and thread based parallel execution of events. The tasks of a Storm streaming application run in different processes in different nodes. Tasks running in the same process can use the memory to communicate, while others running in different processes utilize networks. The communication algorithms are optimized to consider the task locality in order to improve network and interprocess communications. To test the system we use the SLAM streaming algorithm mentioned above and a simple stream processing application with synchronous and asynchronous processing at the parallel tasks for evaluating the behavior of the broadcasting algorithms with different data sizes and nodes. The tests are conducted on an OpenStack-based cloud test bed and a HPC cluster.

II. Evaluation

We evaluated an initial implementation of proposed improvements on a cluster of four nodes with each node hosting four Storm workers. We observed about 10% reduction of end-to-end latency when the payload size was more than 50KB. Our streaming application consists of a single data source, a single broadcaster, 30 workers and a single gatherer in a DAG as shown below:



III. Conclusion

Our preliminary results in Section II shows that these algorithms are a viable solution for improving latencies for real-time applications. We also observed that there is room for improvement to further reduce latencies and network overhead by incorporating techniques such as memory-mapped file-based communications.

References

[1] Kamburugamuve, Supun, Hengjing He, Geoffrey Fox, and David Crandall. "Cloud-based Parallel Implementation of SLAM for Mobile Robots."

[2] Storm, Apache. "Storm, Distributed and Fault-Tolerant Real-time Computation." (2014)

[3] Pjeivac-Grbovi, Jelena, Thara Angskun, George Bosilca, Graham E. Fagg, Edgar Gabriel, and Jack J. Dongarra. "Performance analysis of MPI collective operations." Cluster Computing 10, no. 2 (2007): 127-143.