GPU Accelerated Graph-Based Anomaly Detection

Ian James Thomas, Dr. Enyue (Annie) Lu, Salisbury University

Abstract

Anomaly detection plays an important role in network intrusion detection systems, as anomalous network data could indicate some type of attack. Interconnected network data can be accurately represented as graphs, a structure which contains a set of vertices and edges. This research explores methods for implementing the SUBDUE graph compression algorithm using NVIDIA CUDA, in order to identify common occurrences in a network data set.

Previous Research

Previous research into this approach to graph-based anomaly detection was performed by Brandon Joyce of the University of North Carolina during the 2016 REU, in which a MapReduce approach was implemented. The MapReduce implementation of the SUBUDE algorithm resulted in a 99.21% sensitivity rate (attacks correctly identified) and 94.65% accuracy rate (identifying attack type). Due to overhead associated with distributed computation used in MapReduce, the run time for the algorithm was poor. The goal of this project is to utilize the massive parallelism of GPU programming in order to increase performance improvements over Joyce's MapReduce implementation.

Star-Graph Model

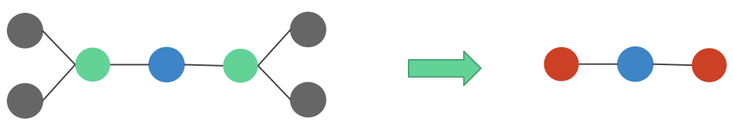
To model network data, a star-graph model is applied. The star-graph model is a way to represent individual network records as a 'hub' vertex, and different attributes of that network record are attached as vertices to the hub. Common patterns in multiple network records are found, then compressed.

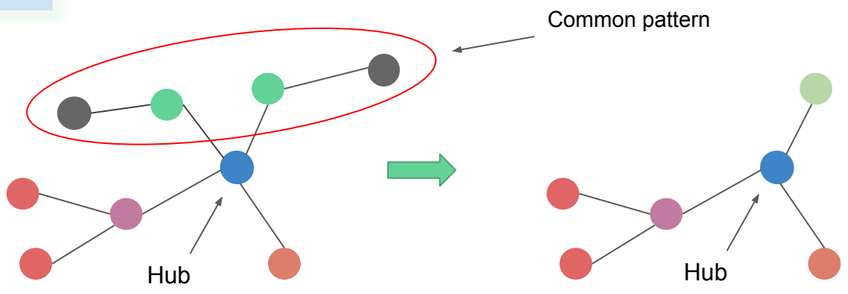
NVIDIA CUDA

NVIDIA CUDA is a parallel computation platform developed by NVIDIA corporation. It allows for CUDA enabled graphics processing units (GPUs) to be utilized for general purpose processing. CUDA enabled GPUs use many parallel processors grouped into streaming multiprocessors which are capable of running concurrent thread blocks. This allows for massive use of parallelization to handle large amounts of data.

SUBDUE Graph Compression

In the SUBDUE graph compression algorithm, common patterns of a graph are found and compressed. This is done by replacing entire subgraphs that occur multiple times with a single node. Typically, graph compression algorithms are used for data compression. While this is useful in our case, as the amount of records we will be processing are massive, it is more useful for tracing how many times a subgraph appears in a data set.





$$A = 1 - \frac{1}{n}\sum_{i=1}^{n}(n - i + 1) * c_i$$

Data Set

The KDD CUP-1999 data set is used in testing, and is the data used for the performance benchmark. The data set is comprised of 5,000,000 records, each containing 41 properties. The data set is nine weeks of simulated air force network traffic. Records are listed as either normal, or some attack type is given

Performance Results

The following are results from Joyce's MapReduce implementation of the SUBDUE algorithm, and results from the pairing and compression phase from our CUDA implementation. Joyce's algorithms were only run once on a 16-node cluster at University of Maryland, Baltimore County's High Performance Computing Facility. The results from our CUDA algorithm were run on one machine

Testing Configuration

System Specification for CUDA implementation (one machine):

CPU: Intel Xeon E5-2650 v4 @ 2.20GHz
GPU: NVIDIA Quadro M2000 (NVIDIA Maxwell Architecture)

Hardware specifications for 16-node cluster running MapReduce algorithm were not listed

# of records processed	Hadoop MapReduce	NVIDIA CUDA (device code only)	NVIDIA CUDA (device + host)
50 records	1,244.25 seconds	.001043 seconds	.0107 seconds
2550 records	Not Tested	.00707 seconds	.4139 seconds
5100 records	Not Tested	.0341 seconds	.7676 seconds
45,214 records	2,011.65 seconds	.3845 seconds	6.574 seconds
667,370 records	7,927.97 seconds	15.425 seconds	73.142 seconds
1,001,289 records	8,365.27 seconds	26.301 seconds	107.109 seconds

Ranking System

A ranking system has yet to be implemented, which would conclude the implementation. This ranking system would consider the number of properties compressed in comparison to how many times that property appears in the record to determine the likelihood of the record being anomalous. This will most likely be easy to implement, and (theoretically) does not affect the performance of the algorithm, as it would run O(r) where r is the number of records, while the compression phase of the algorithm runs O((P*R)^2), which is the most intensive phase of the algorithm. The following equation was used in the original MapReduce implementation, where *n* is the number of iterations, *i* is the current iteration, and *c,* is the percent of the record that was compressed

Conclusion & Future Work

Our results indicate a significant performance improvement when processing the records on the GPU rather than distributed over a cluster. Once the ranking system is complete testing will be done to confirm the same accuracy as the original MapReduce implementation, and testing on other data sets will be done. Future work could include comparing the results to different clustering methods explored in other research.