Investigating and Modelling Scalability for Graph Analytics

Kenrick Fernandes (kenrick@cs.pitt.edu), Rami Melhem
Mohammad Hammoud {Computer Science, Carnegie Mellon University Qatar}

Motivation

Big graph-structured data is ubiquitous – examples include social networks, road networks, World Wide Web links, and biological graphs. A number of graph analytics algorithms are used to analyze this data such as clustering, shortest path analysis, K-core decomposition, node centrality and Google's PageRank.

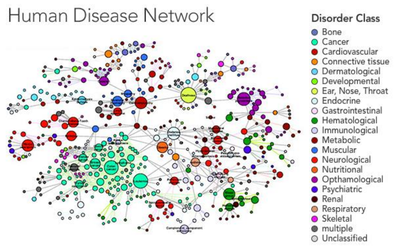

Figure: Node Centrality for disease networks

Figure: PageRank for web graphs

Commodity clusters have become popular platforms for running these analytics algorithms due to the storage space and processing time required to analyze real-world graph data. However, a big barrier to accessibility in using these systems is the cost of finding the right cluster sizes to handle the workload.

In this work, we collect performance measurements for a number of key analytics algorithms on real world graph datasets, model their scalability and predict optimal cluster sizes accurately.

Background

Pregel was one of the first systems to provide a simple programming model for enabling distributed computation on large graphs. The Bulk Synchronous Parallel (BSP) model consists of a round of computation and messaging, called an iteration or *superstep*, and then synchronization at a barrier. This repeats until convergence. Each *superstep* applies a user-defined computation function on the vertices after gathering the messages from the previous superstep. Then messages from the current superstep are scattered.

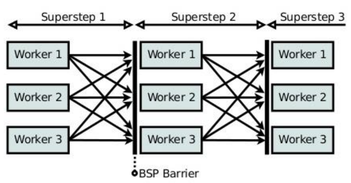

Figure: BSP Computational Model

Apache Giraph is a popular open-source Pregel-like graph processing system, started at Yahoo and incubated at Apache. It is a stable and flexible system built and tested in a production environment, unlike other research prototypes. Giraph inter-operates with other components of the Hadoop ecosystem and relies on Hadoop to access cluster resources. Hadoop's scheduler provides task slots called "workers", to which sections or partitions of the data and computation are assigned.

The flow of control in Giraph, shown on the right, follows a series of steps. First the graph dataset is read in across the cluster, and the computation is performed in parallel. Once the algorithm converges, the output is written back to the distributed file system in parallel across the cluster.

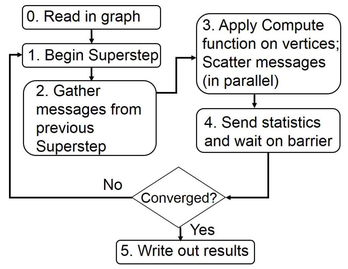

Figure: Giraph Superstep Structure

Characterization and Modelling

We want to answer: how important is resource X for distributed graph data analytics on a production system (Giraph) in a cloud deployment? Here resources include hardware and software resources such as working memory, CPU threads, network bandwidth and memory optimizations. We answer these questions by collecting over 7500 performance measurements from the different classes of analytics algorithm behavior we observed:

Stationary: identical or nearly identical computation and communication load across supersteps (PageRank-**PR** and Diameter Estimation-DE)

Non-Stationary: algorithms in which computation and messaging loads vary across supersteps, for reasons such as traversal of graph structure (Single Source Shortest Path-**SSSP**, Multiple Source Shortest Path-MSSSP) and repeated rounds of sequences of different supersteps (Graph Coloring-GC).

We use 4 graph datasets representing a range of different edge distributions (such as power law and road network), densities and sizes (from millions to billions of edges). More details are in our paper.

We studied the algorithms we chose in greater detail, looking at workloads across supersteps and the impact of scaling cluster size across the workload.

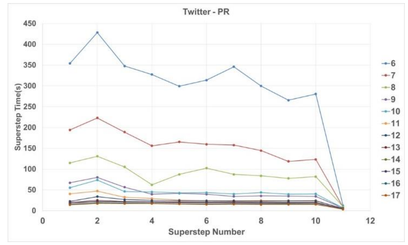

Figure: **PR**– Scaling Impact

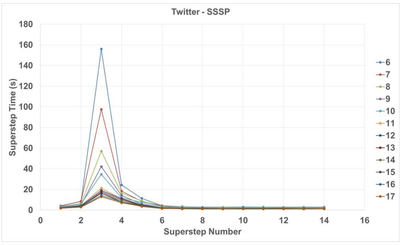

Figure: **SSSP** – Scaling Impact

We observed that the benefits of additional resource allocations from scaling depend on the workload pattern of the algorithm itself and show decreasing returns to scale. We also realized that a range of cluster sizes provide similar performance but with different cost, and so our objective should be to predict the optimal range, rather than a single "sweet spot" correctly.

We create and validate a statistical model for predicting time taken by an analytics job. Our linear model uses non-linear features and includes terms for computation, communication, algorithm workload, impact of graph dataset skew, memory optimizations, and graph partitioning parameters. We build an interpretable model using domain knowledge and our hypothesis that algorithms scaling is based on the graph dataset Skew (difference between average and maximum degree), as well as an algorithm-specific workload pattern.

The model terms account for:

- Communication between vertices, impacted by partitioning, network bandwidth, graph skew and cluster size
- Computation on vertices, impacted by skew, number of threads and cluster size
- Algorithm workload pattern
- Impact of networking optimizations
- Impact of increased graph partitions counts over a baseline that depends on the number of threads

We validate the model by performing feature selection on our dataset of performance measurements. To check if our proposed model terms have explanatory power, we examine models selected via forward stepwise regression and the Akaike Information Criterion. This is an information-theoretic criterion that rewards goodness of fit measured using likelihood. Finally, we train the model using bootstrapping to obtain confidence intervals and observe a low training and 10-fold cross-validation error.

Intepretation and Prediction

The learned coefficients for our model show us that:

- Algorithm behavior has the most impact, with stationary algorithms behaving similarly, but non-stationary algorithms showing a broad variety of behaviors.
- Computation and communication have the most impact on scaling after algorithm behavior, however, the magnitude of their impact is much lower.
- Memory optimizations and increased partition counts have little impact.

To test our model's ability to generalize and predict accurately, we train it on data for a 36-machine virtual cluster and test it on a different 20-machine virtual cluster. We use the model to predict the "sweet range" for cluster size, and we see that it predicts accurately even for settings that it has not seen at all in the training data, such as different numbers of threads.

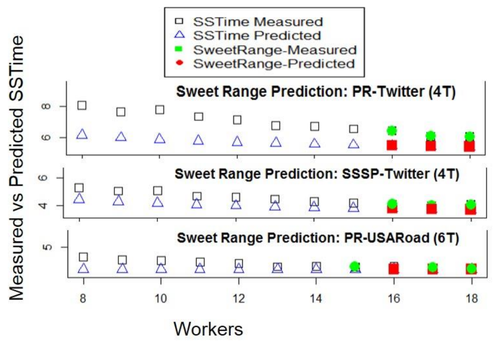

Figure: Accuracy of our model when predicting optimal cluster range

Related and Future Work

Most existing work analytics algorithms focuses on solving problems at a scale which only large technology companies like Google, Facebook and Yahoo face regularly. However, recent industry surveys show that the much more common use cases involves smaller cluster sizes, typically less than 50 nodes, and a few terabytes of data – here resource efficiency is crucial and this is what motivated our work.

Some previous works in the area compare distributed graph processing systems and attempt to evaluate their scalability for specific tasks. However, they do not model or predict performance scalability, which is our focus.

In future work, we aim to increase the breadth and depth of the settings for collecting measurement data and implementing our model in an intelligent system that can make dynamic resource allocation decisions.

*A paper about this work is under submission to IEEE International Conference on Cloud Computing 2018.