The Constellation Project: Representing a High Performance File System as a Graph for Analysis

Andrew W. Nash
Department of Electrical Engineering and Computer Science
Tickle College of Engineering
University of Tennessee
Knoxville, TN 37996
anash4@vols.utk.edu

Abstract

The Titan supercomputer utilizes high performance file systems that change significantly as scientists run simulation algorithms that generate and modify millions of files within a short period of time. The metadata of the files, applications, jobs, groups, and users on these file systems are a rich source for data analysis to extrapolate similarities between the various entities with modern graph algorithms. Since a single snapshot of the metadata is significantly large, an efficient graph library must be utilized in order to perform analysis in real time. This project in lieu of thesis (PILOT) examines the Constellation graph library and implements graph analytics algorithms, including PageRank and SimRank, so that a user can rank vertices and find patterns among the graph efficiently. Results from the analysis are examined to determine if importance in the graph correlates to power users of the system in a given period of time.

1 Introduction

Scientists utilize leadership-class supercomputers to run simulations of complex phenomena by harnessing the parallel processing capabilities of these systems. Since these users are of different fields of study, the high performance file system supporting the supercomputer will contain data from many subject areas, which leads to a rich data source for ranking popular areas of study while exploring similarities between users. Since a simulation is capable of millions of calculations and steps in a short period of time, the contents of a high performance file system change constantly, and logs of these systems can grow significantly large. Therefore, in order to analyze the file system, an efficient method must be employed.

1.1 Titan

The Titan Cray XK7, managed by the Oak Ridge Leadership Computing Facility, is a supercomputer with a peak performance of 27 petaFLOPS and was ranked as the third fastest in the world on the *Top500* June 2016 benchmark list [1]. Titan has been used for research in areas that include: 3D neutron radiation transport for reactors, chemical sciences, climate change science, combustion science, condensed matter physics, electronic structure materials science, and molecular science. Simulations of diseases, weather patterns, and supernovas are just some of the programs that are run on Titan [2]. Atlas1 is a high performance file system that efficiently manages data on Titan. With over 1,000 object storage targets, Atlas1 has a capacity of 14 petabytes [3]. Therefore, the metadata alone is a rich source of information that can be used for graph analytics to attempt to extrapolate relationships between objects and entities in the system.

1.2 Data

The data analyzed for this specific project consists of two snapshots of Titan's high performance file system. The first snapshot was taken on 20 July 2015, and the other is from 21 July 2015, which shall be called *J20* and *J21*, respectively. These snapshots contain lists of the following entities: applications, files, groups, jobs, and users. A breakdown of the quantities of each entity for each date, along with the size of the text file holding the data, is included in the table below.

Entity Type	J20 Quantity	J20 File Size	J21 Quantity	J21 File Size
App	0	303 KB	13	472 KB
File	187,325,446	39.09 GB	187,754,436	39.17 GB
Group	11,505	728 KB	11510	729 KB
Job	600	48 KB	859	67 KB
User	12,987	658 KB	12990	658 KB
TOTAL	187,350,538	39.09 GB	187,779,808	39.18 GB

1.3 Constellation Data Service

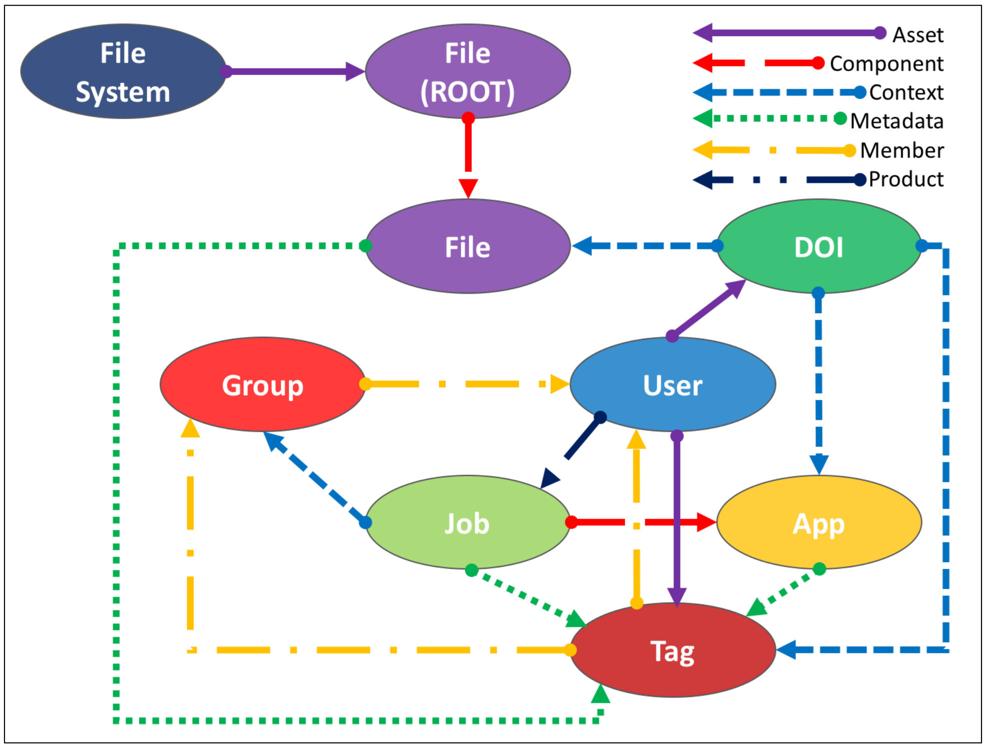

Figure 1: Visual representation of the default graph generated by the Constellation Data Service

Considering that each snapshot is almost 40 GB in size, in order for the data to be effectively analyzed, an efficient graph framework must be utilized. For this purpose, the Technology Integration Group at Oak Ridge National Laboratory developed the Constellation Data Service (CDS) [4]. This program is capable of extracting entities from the snapshot files and generating a graph that can be loaded into a high performance system's memory for analysis. Each entity in the snapshot is represented as a vertex, and the program automatically adds appropriate edges. There are different types of edges, as outlined in Figure 1, to add context to the relationship between two vertices in the CDS. When extracting the entities from the snapshot for each vertex, the following attributes from the metadata of each entity are saved into the vertex structure in the graph:

- App: ID, start time, stop time, exit code
- File: create time, modify time, name, owner user ID, owner group ID, mode

- File System: name
- Group: ID, name
- Job: ID, name, host, start time, stop time
- Tag: name, description, access
- User: ID, user name, legal name, email address

An entity's ID attribute listed above refers to the system-assigned identification number in the high performance file system, which is separate from the index number assigned to each vertex as it is loaded into the graph for traversal purposes. In order to have an efficient graph, some unnecessary metadata provided in the snapshot is not saved. Once the snapshots are loaded into a CDS graph, the state of the graph in memory can be saved to a file for later analysis without having to extract the data from the snapshot again. The graph file of the J20 data set generated by the CDS is 9.82 GB, and the graph file for the J21 data set is 9.85 GB, which is basically one-fourth the size of the original snapshot. This reduction in size is due to efficient compression techniques used by the CDS and the exclusion of unnecessary metadata stored in the snapshot.

2 Implementation

In order to implement graph analytics capabilities into the Constellation Data Service, two modules were created: *CDSAnalytics* and *DATAnalytics*. The former takes a graph file generated by the CDS as input in order to perform analysis that requires traversal of the graph. The latter takes matrices generated by CDSAnalytics as input to perform graph algorithms that require efficient, parallel processing.

2.1 CDSAnalytics Program

The CDSAnalytics program starts by loading a graph file generated by the CDS into the memory of a high performance system. Once the graph is loaded, CDSAnalytics offers additional graph manipulation features. One notable feature provided by CDSAnalytics is the assignment of a vertex key, which shall be called a *VKey*. While the CDS assigns a vertex number to each object to allow for fast traversal, the vertex key assigned by CDSAnalytics is a unique string that can be utilized to find and retrieve a vertex. For all objects except files, this key remains constant and is based on the object's system metadata listed in the snapshot. For example, if an Atlas1 user has the system-assigned ID number 100, that user's vertex key will always be "u100" for every snapshot that features the same user. For files, since Atlas1 does not assign a unique ID number to them, the vertex key relies on the CDS vertex number, which could change between different snapshots. Furthermore, a vertex's type can be easily identified based on the first letter of its key. Below is a table listing the format of each vertex key by type.

Vertex Type	First Letter	+	Example
App	a	System ID	a100
DOI	d	System Number	d9999
File	f	CDS Vertex Number	f10
File System	s	System Name	satlas1
Group	g	System ID	g100
Job	j	System ID	j100
Tag	t	System Name	tclimate
User	u	System ID	u100

Below is the full list of features implemented in CDSAnalytics, listed by the name of each command.

- Edges: Adds asset-type edges from file vertices to the respective user and group vertices that own the file, or adds metadata-type edges from file vertices to the respective user and group vertices, depending on the choice of the operator. This feature is needed for PageRank.
- Export: Exports the graph to a comma-separated value (CSV) file as a list of edges, where each line in the resulting file represents an edge. Each line is in the following format: `source_vkey,destination_vkey,edge_type`

- Index: Generates an index that maps a vertex's VKey to a pointer that indicates the location of the vertex in memory to allow for an extremely fast query. However, this comes at the cost of more memory usage on the system.
- Modify: For every file vertex in the graph, an edge is added from the user vertex that owns the file to the group vertex that owns the file, if such an edge does not already exist. This feature is needed for SimRank.
- PR: Generates a file containing the matrix necessary to run the PageRank algorithm.
- Print: Prints the first level of file vertices that are children of the root file.
- Query: Queries a vertex by its VKey to retrieve all of its metadata stored in the graph.
- SR: Generates a file containing the matrix necessary to run the SimRank algorithm.
- Store: Store the current state of the graph to a file so that it can be reloaded later.
- Total: Shows total number of the vertices, broken down by each type.

2.2 DATAnalytics Program

Since the graph takes up a significant amount of memory when loaded, the DATAnalytics program was developed as a separate module so that it could be run without having the graph itself consume memory. It relies on matrix files generated by CDSAnalytics as input.

2.2.1 PageRank

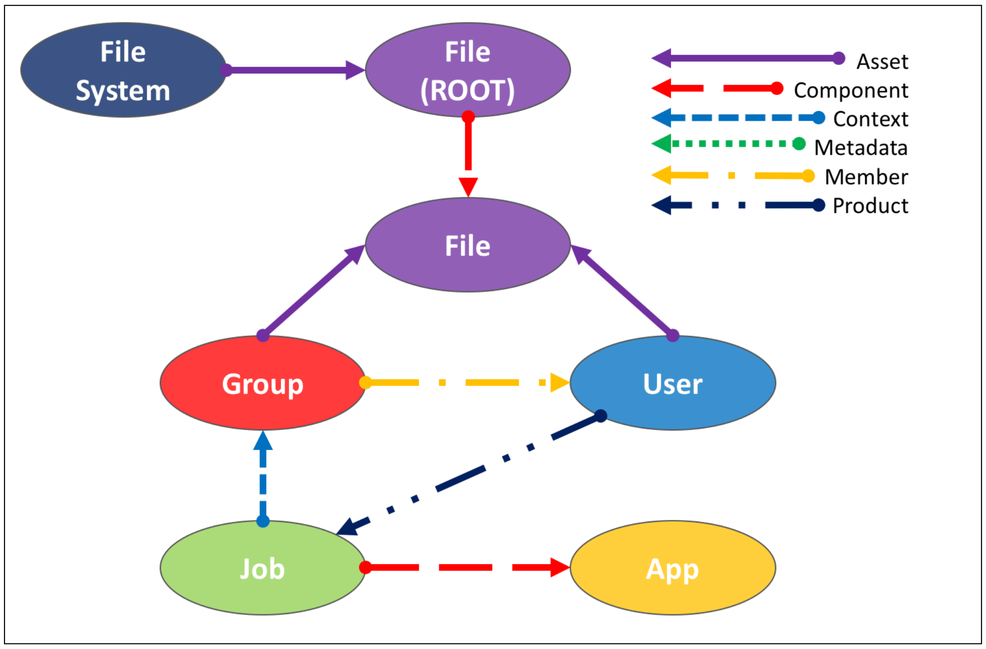

Figure 2: Modified graph to represent files as assets of the groups and users that are owners

Google's PageRank [5] is well known for its effectiveness in searching for relevant content on the Internet. Therefore, it was the first algorithm implemented into DATAnalytics to attempt to rank the vertices. The PageRank technique presented by Austin [6], where the highest-ranked vertices have high-ranking vertices pointing to them, was utilized. However, since, as shown in Figure 1, the default configuration of the graph generated by the CDS does not have edges that directly connect file vertices with user and group vertices, modifications had to be made to the graph in order to generate meaningful results, since the J20 and J21 data sets did not have tag nor DOI entities included. The Edges function in CDSAnalytics was utilized to generate two different graphs. The first modified graph, shown in Figure 2, has asset-type edges from group and user vertices to all of their respective files to indicate ownership. This modification was made to attempt to find important files in the graph. The second modified graph, shown in Figure 3, has metadata-type edges from file vertices to the group and user vertices that own them. The graph was created to attempt to rank users and groups based on the number of files each owned in the file system.

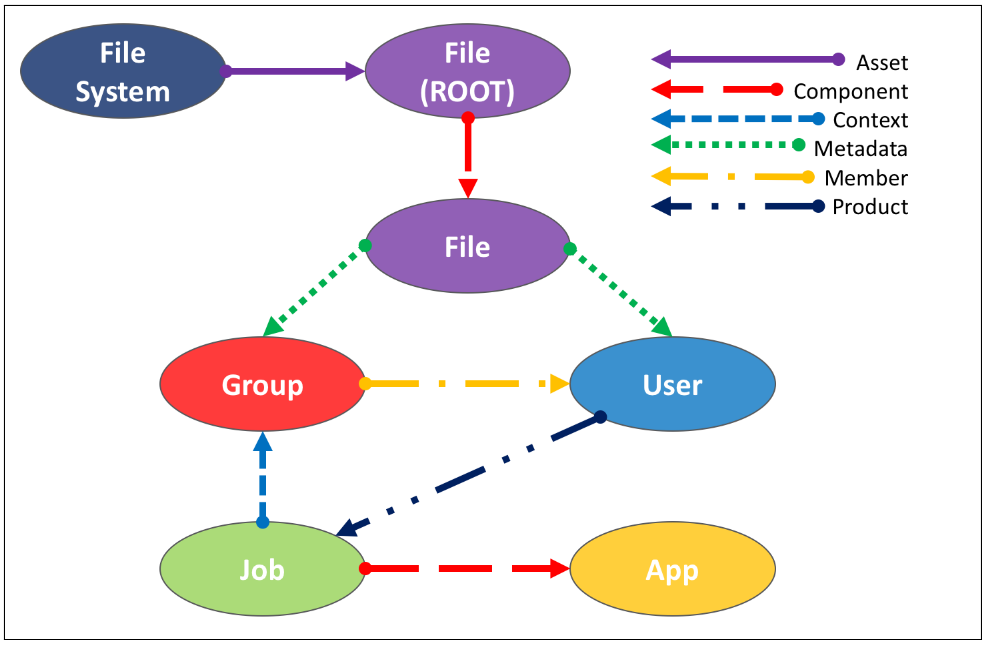

Figure 3: Modified graph to represent files as metadata leading to groups and users that are owners

The PageRank algorithm in DATAnalytics takes a matrix file generated by CDSAnalytics as input. This matrix is the adjacency matrix on which PageRank relies and is set up as follows: if there are n vertices in the graph, then the corresponding adjacency matrix H will have n rows and n columns, and each vertex will be assigned an index from 1 to n. Then if the vertex at index j has an edge leading from it to the vertex at index i, then the entry at $H_{i,j}$ will be $1/x$, where x is the number of edges leaving the vertex at index j. All of the elements in a given column should sum to 0 or 1. For example, the adjacency matrix of the graph in Figure 3 is:

$$\begin{array}{c c} & \begin{array}{ccccccc} System & Root & File & Group & User & Job & App \end{array} \\ \begin{array}{c} System \\ Root \\ File \\ Group \\ User \\ Job \\ App \end{array} & \left(\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 1/2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 0 \end{array}\right) \end{array}$$

Since the adjacency matrix is sparse, consisting mostly of zeros, DATAnalytics efficiently saves the matrix in memory by utilizing three vectors, one each for the row indices, column indices, and values in H, in order to only store the non-zero values, along with their respective indices. Then, DATAnalytics, containing the PageRank algorithm that utilizes the power method as outlined by Austin [6], calculates the PageRank of the graph with a user-specified damping factor. Since the PageRank algorithm consists of a series of matrix and vector multiplications, DATAnalytics performs calculations in parallel, resulting in significant speedup. As a result of the parallel processing and efficient sparse matrix storage, DATAnalytics is capable of calculating the PageRank of the J20 and J21 data sets within a couple of hours, with the iteration tolerance set to 0.001.

2.2.2 SimRank

The SimRank algorithm [7] was designed to rank the similarity between two given vertices by assigning a value in $[0, 1]$ to score the relationship. A value of 1 indicates that the two vertices are the same, while a value of 0 indicates no similarity. It is so effective, Twitter staff implemented a variant of SimRank to compare two users for their recommendation engine [8]. This algorithm takes the same adjacency matrix that PageRank uses as input, and the result is a symmetric matrix containing the similarity scores between two objects, with all elements in the diagonal equal to 1.

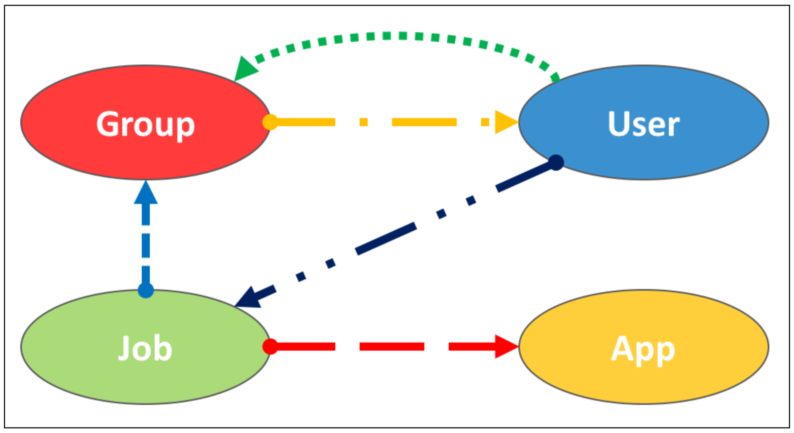

Figure 4: Modified graph with files replaced by edges from the owning user to owning group

SimRank relies on a series of matrix multiplications, which makes it computationally intensive for large matrices. Therefore, it is impractical to calculate the SimRank of the entire J20 and J21 data sets. Since this project is primarily focused on finding similarities between users and groups, the Modify feature was implemented in CDSAnalytics to generate a much smaller adjacency matrix. For every file vertex in the graph, an edge is added from the user vertex that owns the file to the group vertex that owns the file, if such an edge does not already exist. Then, the file vertices are removed, and the resulting adjacency matrix is generated for use in SimRank. The graph in Figure 3 would thus be modified to that shown in Figure 4. While the matrices start as sparse, they become more dense after each iteration. Therefore, in order to maintain efficiency, the Boost uBLAS library [9] was utilized to store the matrices as mapped matrix structures to conserve storage and ignore numerous multiplication operations with zero. Furthermore, matrix multiplication operations were done in parallel. DATAnalytics utilizes the SimRank algorithm as outlined by Antonellis et al. [10].

```
Algorithm SimRank
Require: Adjacency matrix $P$, Decay factor $decay$, Number of iterations $k$
  $S \leftarrow I$
  for $i = 1 : k$ do
    $T \leftarrow decay * P^T * S * P$    // parallel operations
    $S \leftarrow T + I -$ Diag( diag( $T$ ) )    // parallel operations
  end for
```

3 Results

The CDSAnalytics and DATAnalytics programs were run on multiple high-performance systems that each contained approximately 96 GB of memory. Each system utilized 8 cores to allow for a peak of 800% CPU usage while performing parallel matrix operations. Since multiple systems were used, the time to completion for each algorithm was not recorded. However, all processes completed within approximately four hours of wall time, indicating that these algorithms could be reasonably implemented into the CDS and periodically run as batch jobs to monitor graph analytics or power a search engine.

3.1 PageRank

The PageRank algorithm was run with three different damping factor values α: 0.15, 0.50, and 0.85. Furthermore, the algorithm was tested on four different graphs: J20 and J21 with asset-type (A) edges as shown in Figure 2, along with J20 and J21 with metadata-type (M) edges as shown in Figure 3. The top groups and users with the highest PageRank scores–along with the respective pseudo-IDs that indicate the types of vertices, number of edges entering each, and number of edges leaving–are included in the following table.

Data	α	ID	Subject	In	Out	Notes
J20 A	0.15	G94	Biophysics	122	7967	Biophysics 2nd, 3rd
J20 A	0.15	U67	Comp. modeling	14	1	
J20 A	0.50	G94	Biophysics	122	7967	Biophysics 2nd, 4rd, 7th
J20 A	0.50	U05	Scientific comp.	16	41,324	
J20 A	0.85	G94	Biophysics	122	7967	Biophysics 2nd, 3rd
J20 A	0.85	U67	Comp. modeling	14	1	
J21 A	0.15	G73	Biophysics	119	11	Biophysics 2nd, 3rd, 9th
J21 A	0.15	U67	Comp. modeling	14	1	
J21 A	0.50	G73	Biophysics	119	11	Biophysics 2nd, 3rd, 10th
J21 A	0.50	U67	Comp. modeling	14	1	
J21 A	0.85	G73	Biophysics	119	11	Biophysics 2nd, 3rd, 10th
J21 A	0.85	U67	Comp. modeling	14	1	
J20 M	0.15	G70	Computer science	6	9	
J20 M	0.15	U11	Computer science	12,417,090	2	G70 leader
J20 M	0.50	G70	Computer science	6	9	
J20 M	0.50	U11	Computer science	12,417,090	2	G70 leader
J20 M	0.85	G70	Computer science	6	9	
J20 M	0.85	U52	Computer science	5	0	G70 member
J21 M	0.15	G80	Climate	1,098,650	28	Climate 10th
J21 M	0.15	U32	Computer science	3043	0	
J21 M	0.50	G80	Climate	1,098,650	28	Climate 7th, 10th
J21 M	0.50	U32	Computer science	3043	0	
J21 M	0.85	G96	Nuclear physics	256,657	2	G80 3rd
J21 M	0.85	U70	Accelerator physics	61,347	2	

As shown in the table, the change in damping factor had minimal effect on the PageRank calculations with the graph. While groups related to biophysics had high scores in the graphs with asset-type edges leading to files, running PageRank on that graph did not yield meaningful results since most edges were distributed. As a result, many files in this type of graph had the same PageRank values, leading to difficulty in identifying importance. However, with the graphs that have metadata-type edges, the results are quite interesting since the edges are clustered around specific groups with high activity. Note that in the J20 M data, a user, U52, with few files received the highest PageRank score due to working closely with a power user in the same group, U11. Many of the files with the highest ranks in this graph were home directories for various users and groups. Based on preliminary results, PageRank has significant potential to allow researchers to identify constellations among users of supercomputing systems by ranking the objects on the system. This could ultimately be used to build a search engine that returns results based on activities and connections.

3.2 SimRank

Since SimRank, even with a parallel implementation, is computationally intensive, 5 iterations were chosen, along with the standard decay factor of 0.80. Since SimRank compares two vertices, some of the top users and groups according to PageRank were examined to find vertices that were most similar to them. Furthermore, a few pairs of vertices with the highest SimRank scores of 0.8 were examined. The results are shown in the following chart:

Data Set	ID	Subject	Best match	Match subject	SimRank Score
J20	U11	Computer science	G70	Computer science	0.043
J20	U85	Staff	Multiple	Staff	0.800
J20	G29	Staff	U91	Staff	0.109
J21	G96	Nuclear physics	G02	Physics	0.518
J21	U70	Accelerator physics	G89	Accelerator physics	0.152
J21	G96	Nuclear physics	G22	Computer science	0.518
J21	G96	Nuclear physics	G75	Physics	0.259

A high SimRank score did correspond to similarities in the actual activities and subject areas of the users.

- U11 is the primary scientist in the G70 research group. See the PageRank scores.
- U85 is a generic user account that is utilized by students in a classroom setting. SimRank accurately identified it has being very similar to all of the other generic student user accounts on the system.
- G29 is a group of staff system engineers that maintain the supercomputer. SimRank identified U91 since that user is a staff member in the group and is a power user.
- G96 is a physics research group and G02 is the default group of a physic researcher.
- U70 is one of the leading accelerator physics researchers responsible for the G89 group.
- G22 is the default group of a user that is a researcher in the G75 group, and since the research in G96 and G75 is similar, G96 and G22 are also related.

The SimRank scores are quite useful in identifying similarities between actual users and groups of the high performance file system, as most high scores between vertices did correlate to common research areas. The capability of SimRank could be expanded through the use of tag vertices, where tags are subject areas connecting respective users and groups. Additional modification to the graph would lead to even more useful SimRank results.

4 Conclusion

Since the data sets are so large, the Constellation Data Service does an effective job of efficient memory management while allowing for fast traversal of the graph. The introduction of the VKey in CDSAnalytics gives algorithms and users the ability to find a vertex in the J20 and J21 data sets quickly by utilizing a mapped index that can fit into the memory of a high performance system. CDSAnalytics is proven to be capable of efficiently generating an adjacency matrix for DATAnalytics. Even though the adjacency matrix used for SimRank did not contain the files in the data set, the similarity scores between users and groups correlated with actual similarities in research areas. Therefore, placing a greater emphasis on users and groups, while considering quantity of file connections, can lead to identifying constellations more quickly.

The SimRank and PageRank calculations implemented in DATAnalytics generated useful results that can be utilized to rank and compare users of a high performance file system. The algorithms are computationally demanding for large data sets, thus requiring efficient parallel performance and memory usage. Since CDSAnalytics and DATAnalytics were written as modules that can be customized, these programs could be integrated into the CDS itself. The CDS could also automatically perform system calls to CDSAnalytics and DATAnalytics to generate new results, either in set time intervals or once a certain threshold percentage of the graph has been modified. Regular PageRank and SimRank comparisons of the graph snapshots could identify the hot spots of high activity. These identified hot spots could then be compared with Titan usage logs to determine a correlation.

Additional modifications could be made to the graph algorithms to generate custom results. For example, the various edge types could be weighted differently, or groups in the same area of research could be directly linked with tag vertices. While these graph algorithms are a solid foundation for analysis of the file system, implementation into a production CDS environment could yield additional insight into user behavior and allow for real-time analysis that would be helpful to high performance file system engineers and administrators. Furthermore, new experimental algorithms could be tested to attempt to extrapolate relationships among all of the file system entities more efficiently. This experimental graph research of a high performance file system could ultimately lead to better collaboration among researchers of various scientific disciplines and even more efficient utilization of limited high performance computing resources.

5 Acknowledgements

This Master of Science project in lieu of thesis (PILOT) is based on the Constellation Project, which is an initiative led by the Technology Integration Group at Oak Ridge National Laboratory (ORNL). Group members who contributed resources that this PILOT utilized include: Group Leader Sudharshan S. Vazhkudai, John Harney, Raghul Gunasekaran, Dale Stansberry, Seung-Hwan Lim, and Tom Barron. This PILOT relied on data and resources provided by the Oak Ridge Leadership Computing Facility, managed by UT-Battelle for the Office of Science of the United States Department of Energy. Oak Ridge Associated Universities provided the means for collaboration with ORNL staff. Committee members overseeing this PILOT include: Major Professor Michael W. Berry, Audris Mockus, and Arvind Ramanathan. Thank you very much for your assistance and collaboration.

References

[1] E. Strohmaier, J. Dongarra, H. Simon, and M. Meuer. TOP500 Supercomputer Sites. http://www.top500.org, 2016.

[2] Titan Cray XK7. https://www.olcf.ornl.gov/computing-resources/titan-cray-xk7/.

[3] Lustre Basics. https://www.olcf.ornl.gov/kb_articles/lustre-basics/.

[4] S. S. Vazhkudai, J. Harney, et al. Constellation: A Science Graph Network for Scalable Data and Knowledge Discovery in Extreme-Scale Scientific Collaborations. In *IEEE Workshop on Big Data Metadata and Management*, 2016.

[5] S. Brin and L. Page. Reprint of: The anatomy of a large-scale hypertextual web search engine. In *Computer networks* 56(18):3825-3833, 2012.

[6] D. Austin. How Google finds your needle in the webs haystack. In *American Mathematical Society Feature Column* 10:12, 2006.

[7] G. Jeh and J. Widom. SimRank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

[8] A. Goel, A. Sharma, D. Wang, and Z. Yin. Discovering similar users on Twitter. In *11th Workshop on Mining and Learning with Graphs*, 2013.

[9] Sparse Matrix. http://www.boost.org/doc/libs/1_42_0/libs/numeric/ublas/doc/matrix_sparse.htm.

[10] I. Antonellis, H. Garcia-Molina, and C. Chang. Simrank++: query rewriting through link analysis of the click graph. In *Proceedings of the VLDB Endowment* 1(1):408-421, 2008.