2017 IEEE 24th International Conference on High Performance Computing (HiPC)

Applying Graph Analytics to Understand Compute Core Usage and Publication Trends in a Petascale Supercomputing Facility

Sangkeun Lee, Sudharshan S. Vazhkudai, Raghul Gunasekaran
Oak Ridge National Laboratory
{lees4, vazhukudaiss, gunasekaranr}@ornl.gov

***Abstract*—The Oak Ridge Leadership Computing Facility (OLCF) runs Titan, the No. 4 supercomputer in the world, to deliver over four billion compute core hours every year to several scientific domains, in their pursuit of leadership science. In this paper, we analyze four years worth of heterogeneous log data sources from the OLCF resource fabric, capturing metadata on entities such as users (2,546), scientific project allocations (674), jobs (1,352,402) and publications (1,146), to derive insights into the trends in core hour usage and publications, across 35 science domains. We have constructed a scalable graph to represent the OLCF entities and apply rich graph analytics for our analysis. Based on this, we have analyzed the metadata across five dimensions, namely (1) quantitative analysis of Titan system usage, (2) quantitative analysis of OLCF publications, (3) correlation analysis between system usage and publications, (4) text analysis to derive OLCF research trends, and (5) utilization of graph mining for association analysis. To the best of our knowledge, our work is the first of its kind to apply graph-based big data techniques to provide comprehensive insights on an HPC center's core hour usage and users' publication trends. Our results provide valuable details into an HPC center's core allocation program, measuring the productivity of scientific domains, the interplay between core usage and research output, accelerating collaboration, and in predicting new connections between resource entities.**

I. Introduction

The U.S. Department of Energy's (DOE) Oak Ridge Leadership Computing Facility (OLCF) [1] hosts the world's No. 4 supercomputer, the 27 petaflops Titan system that comprises of 18,688 CPU/GPU compute nodes and 710 TB of system memory. Titan caters to a diverse user base from national labs, academia, and industry, in solving grand challenge problems from 35 science domains such as Climate, Combustion, Fusion, Chemistry, Energy Storage, Materials, Biology, Astrophysics, and Nuclear Physics. Each Titan compute node comprises a 16-core AMD CPU and an NVIDIA Kepler GPU, with 14 streaming multiprocessors (SM), for a total of 560,640 cores. User project allocations are charged 30 units per node-hour, accounting for the 16 CPUs and 14 SMs. In addition to Titan, the OLCF hosts several other resources such as the 32 PB Lustre-based parallel file system (PFS), the 60+PB HPSS disk/tape archive, and several analysis and visualization clusters (Fig. 1). Using all of the above infrastructure, Titan delivers over four billion core hours every year to rigorously peer-reviewed scientific user projects.

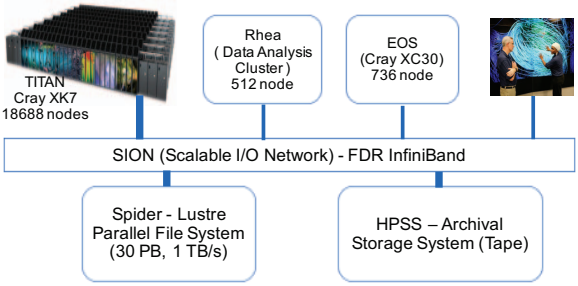


Fig. 1: Overview of systems in OLCF.

A sizeable chunk of Titan's core hours is consumed by leadership jobs, which consume at least 20% of the system and exercise Titan's unique heterogeneous node architecture. Users routinely run several hundreds of thousands of jobs on Titan every year that simulate complex scientific phenomena, and glean insights. Based on such a use of the system, users produce around 250 peer-reviewed publications every year in leading scientific journals and conferences. The scientific productivity of the OLCF national user facility is evaluated based on its successful delivery of computing time to its user base and on the users' ability to conduct science using those core hours, which is mainly measured by publications. It is therefore vital to understand the trends in compute core usage and scientific publications of the OLCF.

Contributions: In this paper, we have conducted an *empirical* analysis of the OLCF's core usage and publication data across several science domains, using four years (2013-2016) of log data. For example, we are interested in studying the trends in job size (both in terms of nodes and hours used) across science domains, trends in monthly core hour utilization, collaboration trends in/across science domains, trends in user participation, etc., which can help the center's

0-7695-6326-0/17/$31.00 ©2017 IEEE
DOI 10.1109/HiPC.2017.00041

resource utilization council to appropriately allocate resources in the future.

HPC center staff often goes to great lengths to track down the papers that used the center resources as it is a crucial metric when it comes to evaluating a center's productivity. In addition to system usage analysis, we are interested in understanding the trends in research publications of science domains based on a quantitative analysis. For example, we have analyzed publications over time, papers per project/user, citations per paper/domain/user, average author counts in publications per domain, and the collaboration metric in publications. Further, we have studied the *correlation* between system usage and scientific publications across the various domains, combining core usage data with publication data.

We have utilized *text analysis* and *association analysis* to unravel deeper insights. We have employed text analysis to understand popular (or trending) research terms (or topics) associated with publications and jobs over time. We have used *association analysis* (graph mining) to derive collaborations of OLCF users (community detection) and speculative relationships between publications and the scientific project allocations they may belong to (predictive analysis). The predictions based on the association analysis can be particularly useful in the absence of clear reporting by users (as is often the case in HPC centers).

To perform such a quantitative, semantic and predictive analysis in a systematic manner, we propose to view the OLCF HPC resource fabric as a graph, and apply a rich set of graph analytics techniques atop. *Graph*-based approaches can analyze heterogeneous information with lots of connections, while also allowing the ingest of new data sources without having to change the database schema.

We have constructed a property graph by representing a diverse set of HPC log data sources over several years, comprising of entities such as users (2,546), projects (674), jobs (1,352,402) and publications (1,146) as graph vertices and the connections between them as edges. We have also derived new connections based on predictive associations between users, projects, jobs, and publications. For instance, we have extracted keywords (also represented as vertices in the graph) from publications using text-processing techniques, and calculate the semantic relevance scores between publications and keywords. The constructed graph is imported into the neo4j [2] graph database and queried using a declarative graph query language called *Cypher* [3] to process various analytics tasks.

To the best of our knowledge, our study is the first of its kind to apply graph-based big data techniques to understand heterogeneous data sets for a leadership-scale HPC center. Particularly, we show (1) how to exploit the constructed graph to correlate publication and job log analysis and (2) to predict missing links between entities, which have not been studied before. Further, our statistical results provide valuable insights into an HPC center's core allocation, measuring the productivity of scientific domains, the interplay between core usage and research output, accelerating collaboration, and in predicting new connections between HPC resource entities. Finally, we expect that our approach of deriving knowledge from heterogeneous data sets will inspire many other system administrators, researchers, and data scientists from other facilities to undertake similar such efforts.

II. OLCF Data Sources

We have used the following OLCF log data sources for the analysis in this paper.

a) User and Project: OLCF maintains and tracks all user activity using a RATS (Resource Allocation and Tracking System) database. The database keeps track of user information, such as name, *username*, affiliation, and contact details; and project allocation information, such as project name, ID, description, compute hours allocated and used. A user can be a part of multiple projects, where a project represents the science domain. For example, Astrophysics projects have a project ID starting with *ast* followed by a number to identify individual projects within the domain. RATS provides APIs to query this information. We have used the RATS data for associating publication and job log datasets (both of these logs are explained below). More specifically, the publication data set has author names and affiliations, which we map to the user's *username* in RATS. Similarly, the job log has only the *username* field to associate to the specific user in RATS, and thereby the associated publication.

b) Job logs: We have used the Titan scheduler's (MOAB) log of job events for our analysis. Each job log entry provides details such as *job-ID*, *username* user who launched the job, *project-ID* the user's project allocation, start time or end time of the simulation job, and the number of compute nodes allocated. A new log entry is created at the start of the job and another log entry at the end of the job. We have used four years of job logs from 2013 to 2016 for the analysis in this paper.

c) Publication: OLCF collects publications that have acknowledged OLCF in the text by using a web-crawler and manual reporting. A web-crawled publication entry needs to be finalized by the OLCF staff. The publication database keeps track of author names, affiliation, associated project ID, title, abstract, type of publication (e.g., book, journal, conference, etc.), name of venue, year, impact factor, citation count, etc. Currently, association of a project ID with a publication is manually processed by the OLCF staff based on user reporting and an expert's classification. However, not all publication entries in the collected log data have been associated with a project. Unlike the job log dataset, collecting a publication entry is a semi-automatic process, and can be potentially incomplete. Therefore, the publication entries should be carefully considered for analysis.

III. Analysis Overview

In this section, we present an overview of our analysis of the compute core usage on Titan and the publication trends. We have employed a graph to represent the entities (users, projects, jobs, keywords and papers) in the OLCF

data sources, and apply graph analytics towards our analysis. We have categorized our analysis tasks into five dimensions, namely (1) Quantitative analysis of Titan system usage, (2) Quantitative analysis of publications produced from Titan usage, (3) Correlation analysis between Titan system usage and publications, (4) Text analysis of publications to derive OLCF research trends, and (5) Utilization of graph mining for association analysis.

A. Quantitative analysis of Titan's compute core usage

The objective of this analysis is to understand Titan's system utilization patterns and usage trends over time by analyzing the job logs. As a first step, we categorized the executed jobs into three different job size bins, namely small (less than 375 nodes), medium (375 to 3750 nodes) and large (greater than 3750 nodes). The next task is to identify and understand the variance seen across science domains, in the utilization of Titan. To this end, we have processed graph pattern matching queries to analyze the graph and interpret the query results as shown in Section IV-C. Such an analysis has enabled us to answer questions like 'Which science domains used the most core hours?', 'Which science domains tend to execute large or small-scale jobs?', 'Can we expect particular science domain's job utilization peaks in a particular period of a year?'. The answers to such questions can provide meaningful insights to HPC centers for future deployments and time allocation.

B. Quantitative analysis of OLCF acknowledged publication

In this dimension, we first analyzed the aggregated statistics on publications (e.g., the number of publications, aggregated citation counts), and then we delved into the variance in different science domains. This task has helped us in answering questions such as 'Which science domain published the most?', 'Which domain has the most publication or citation count per person?', 'How does the number of publications change over time?', etc. Furthermore, we have analyzed the collaboration trends by closely observing authorships to answer questions like 'How does the average number of authors differ in different domains?', 'Which is the pair of domains that collaborated the most?', 'Which domain has less tendency to collaborate with the other domains?'. This analysis has provided insights to OLCF regarding not only the quantity but also the quality of publications. Similar to the previous case, we have employed graph queries for the analysis.

C. Correlation analysis between Titan usage and publications

In this analysis, we have investigated the correlation between jobs and publications, as job executions and publication activities are not independent of each other (e.g., scientists perform jobs on HPC systems, which leads to the publication of academic papers). By studying the correlation, we have answered questions like 'which domain has the highest publication and citations per core hour?', 'what size of jobs has led to more publications or citations?' (i.e., do large leadership jobs necessarily mean more papers?), 'which domains have more users who both use HPC systems and also publish papers?'. By observing these two different data sets together (publications and jobs), we have derived unique insights that cannot be accomplised by any one data set in isolation. For this analysis, graph edges, connecting jobs and publications, played a crucial role.

D. Text analysis of publications for OLCF research trends

A very valuable piece of information in the publication dataset is *text* information such as *title* and *abstract* of the papers. We have extracted keywords from a publication's text and associated the keywords with the publication. We then used a text analysis technique called LSA (Latent Semantic Analysis) [4] to compute the semantic relevance scores between and across publications and keywords. We used the computed scores to assign edge weights between/across *Publication* and *Keyword* vertices in the graph. Based on the graph components constructed from the publication text, we have derived semantic overviews regarding the contents of the OLCF scientific projects with various perspectives. It has provided us answers to questions like 'What are the most popular keywords in publications over time?', 'How do keywords describing executed jobs change over time?', 'Which science domains share similar research topics?', 'How does the prevalence of research related to keyword '*nuclear*' change over time? (e.g., similar to Google Trend)', etc.

E. Utilization of graph mining for association analysis

To derive further insights from associations in the OLCF data sources, we have also employed two graph mining approaches, *community detection* and *link prediction*. For community detection, we have modified the *connected component* concept of an undirected graph to operate with a property graph, and used it to analyze users' collaboration associations. For *link prediction*, we have quantified and utilized relationship strength between two vertices in the graph, considering paths existing between the vertices to find the speculative relationships. Specifically, we have presented a use case of predicting missing edges (*deliverableOf*) between *Project* and *Publication* vertices. This capability is of significant value to an HPC center like OLCF, as it has enabled us to identify and predict publications that were produced by scientific projects, in the absence of proper reporting, which is often the case.

IV. Graph-based Data Analysis Framework

To conduct the above analysis, we have employed a graph-based approach to tie together the entities from the HPC data sources.

A. Why graph?

OLCF data sources are heterogeneous, with many entities in them having connections between each other. Often times, these relationships are not explicitly described in the data sources. For instance, an author of an OLCF publication can be a user who executed a job on an OLCF HPC system. For successful data analysis such as the aforementioned, it is crucial to establish an integrated data analysis framework that

can capture insights hidden in the connected heterogeneous entities. The data analysis framework should not be constrained by a single data source's limitations in the description, but be able to consider them in concert, towards greater flexibility and richer data analysis.

The graph model has been attracting much attention to represent and understand such complex connections across heterogeneous entities [5]. The graph model also allows flexibility, as it is not constrained with a rigid schema. Data can easily grow even when new data sources with different schema and/or additional columns for the existing data sets are continually introduced and ingested, unlike the relational data model. Motivated by these characteristics, we have adopted a graph-based approach. We have constructed a graph out of the available OLCF's data sources, and query the graph to discover knowledge by delving into the existing subgraph patterns in the constructed graph.

Specifically, we have adopted a *property graph* [6], which is a directed, typed, attributed multi-graph that can naturally represent heterogeneous entities as vertices, relationships as edges, and capture their descriptions as vertex and edge attributes.

B. Graph construction

The construction of the graph is a three step process as follows.

1) Graph schema design: First, we define the logical structure of our property graph, namely *graph schema*, by identifying vertex types, edge types and their associated attributes from the data source. Fig. 2 shows the schema of the property graph that we have constructed. For each data source, we have extracted vertices and edges from the data source based on a pre-defined conversion mapping between the schema of the data source and the defined graph schema [7]. A single entry from a data source can be converted into a number of vertices and edges with different types. For instance, we may convert each data entry, e_p, in the publication data set, having relational schema $R_{Publication}$(*DOI*, *title*, *list of authors*, *abstract*, *journal*, *month*, *year*) into a *Publication* vertex, p, a set of vertices, $u_1,\ldots, u_n$, of type *User*, a *Journal* vertex, j, a set of edges of type, *writtenBy*, connecting p and $u_1,\ldots,u_n$, and a *publishedIn* edge, connecting p and j.

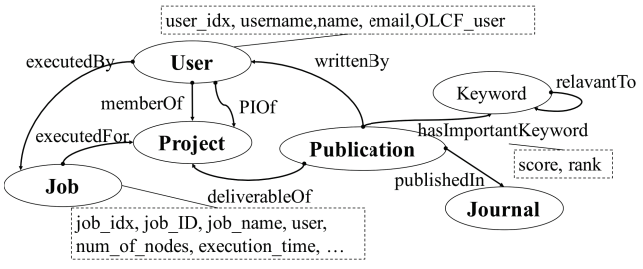


Fig. 2: Overview of the graph schema for OLCF data sources. Note that we only show attribute names for *User*, *Job* vertex type, and the edge type, *hasImportantKeyword*.

2) Merging vertices representing the same entity: Ideally, we do not want to have multiple vertices representing the same entity. However, initially when we extract vertices from disparate data sources, there can be vertices representing the same entity, and we need to merge them as a single vertex. It is straightforward if vertices generated from different data sources use a common identifier (ID), then we can simply merge the vertices with the same ID. However, this is not true for all cases. If common identifiers are not available, we have used a secondary similarity join approach to identify and merge identical vertices.

For instance, the name (i.e., a person's name) property can be used to identify and merge the vertices representing the same person. More specific conditions can be added to increase the accuracy of same entity identification. Generally, instead of using the secondary key value strictly to join, we have computed the Levenshtein distance (d) between the values of the secondary join key and perform similarity join, where we have merged vertices if two vertices are the closest to each other based on d, and d is less than 10% of the average of two join key value lengths. This is useful, because same contents (e.g., person's name) can be described in various ways (e.g., having a full or initial middle name) in different locations. As a result, now the merged vertices become the intermediate connectors on the paths between vertices generated from different data sources (e.g., *(Job)-[executedBy]→(User)←[isWrittenBy]-(Publication)*). This approach has helped to consolidate the graph, and merge identical vertices from multiple data sources.

3) Deriving semantic graph components from text: Next, we have utilized the text information available in the publication data source such as *title* and *abstract* to further enhance the graph and derive new associations between the graph entities. To this end, we have first constructed a term-document matrix by using the text from each publication entry, using the TF-IDF technique (Term Frequency-Inverse Document Frequency) [8], after filtering out stem words. Then, we have performed the LSA (Latent Semantic Analysis) technique [4] which finds a lower-rank approximation of the matrix, using singular value decomposition (SVD) [9] and uses it to achieve semantic similarity between word-document and word-word by computing cosine similarity across vectors in the matrix. We have used this outcome as an additional information to enrich the graph and created *Keyword* vertices, and edges of type, *hasImportantKeyword* (word-document) and *relaventTo* (word-word), which contain similarity scores in their properties. We have used top-30 most relevant keywords for each publication and top-10 most similar keywords for each keyword to create the edges of type, *hasImportantKeyword* and *relaventTo*, respectively. These additional vertices and edges have been useful in understanding the detailed contents of research activities and in predicting missing information in the graph (Section V-E2).

We have developed a tool to process the above three steps, and executed it on a 2.7GHz, 12-Core Intel Xeon E5 machine with 64GB of RAM. Total execution time for graph

construction was approximately 100 minutes.

C. Querying the property graph

TABLE I: Statistical summary of the graph.

Node Type	Node#	Edge Type	Edge#
User	2,546	*(Publication)-deliverableOf→(Project)*	932
Project	674	*(Publication)-hasImportantKeyword→(Keyword)*	34,380
Job	1,352,402	*(Publication)-writtenBy→(User)*	6,873
Publication	1,146	*(User)-memberOf→(Project)*	3,592
Journal	299	*(User)-PIOf→(Project)*	301
Keyword	22,906	*(Publication)-publishedIn→(Journal)*	1,141
		(Job)-executedBy→(User)	1,363,082
		(Job)-executedFor→(Project)	1,363,082
		(Keyword)-relevantTo→(Keyword)	232,370

We have used *neo4j* [2] for loading the constructed graph, which is one of the leading commercial graph database management systems. For graph loading and query processing, we have used the same machine that we used for graph construction. Loading the graph into the database took 553 seconds, and the size of graph database is 812.35MB. Table I shows the statistical summary of the constructed graph. We have composed *Cypher* [3] queries representing our analytics questions, and executed the queries on the graph database. The following is a simple example *Cypher* query (Listing 1) to achieve top 5 pairs of authors from 2013 to 2016 ordered by their co-authorship counts in descending order.

Listing 1: Counting the distinct number of co-authored publications of two users.

```
MATCH (u1:User)<-[:writtenBy]-(p1:Publication),
(u2:User)<-[:writtenBy]-(p1)
WHERE p1.pub_year>=2013 AND p1.pub_year<=2016
RETURN p1.name, p2.name, COUNT(DISTINCT p) AS co_author_cnt
ORDER BY co_author_cnt DESC LIMIT 10
```

The first two lines of the query (Listing 1) describe the graph pattern representing the co-authorship of users. It queries two *User* vertices $u1$ and $u2$, where they both have *writtenBy* edges from the *Publication* node $p1$. The WHERE clause makes the graph patterns more specific (e.g., only querying data from 2013 to 2016), and the RETURN clause finally defines the aggregated information that we want to get from the identified graph pattern instances.

D. Employing graph mining for association analysis

1) Community detection: A connected component is a subgraph in which any two vertices are connected to each other by paths, and the vertex in the connected component is not either directly or indirectly connected to vertices in other connected components. We have extended this to include heterogeneous vertices and edges, and defined a "community" as follows. For a given graph G, a path schema, P (i.e., a sequence of edge types), and a vertex type t, a community is a set of type t vertices in which any two vertices are connected to each other by paths with the given path schema, P, and any vertex in the community is not connected to vertices in other communities by paths with the given path schema, P. For instance, given our graph, a path schema describing co-authorship of two users, *(User)←[writtenBy]-(Publication)-[writtenBy]→(User)*, and a vertex type *User*, a community means any pair of users in it is directly or indirectly connected to each other by following the *co-authorship* paths. We have implemented a community detection algorithm that returns all existing communities in a graph. It iteratively executes Cypher queries to perform depth first traversals, following paths with the given schema, for each vertex to identify the communities. We discuss the result in Section V-E1.

2) Link prediction: Link prediction aims to infer new associations between the vertices. To this end, we have quantified the score of the relationship strength between two vertices, and consider the pair with the higher score to be speculatively linked to each other. For instance, the following queries quantify relationship strength between *Publication* and *Project*, and find Top-5 potentially related *Project*s for each *Publication* in two different ways. The result of queries can be used to predict the *deliverableOf* association.

Listing 2: Cypher queries to predict top-5 potentially associated projects for each publication with missing project information in two different ways.

```
-- Query 1
-- Predicting a Publication p1's associated Project
-- using Publication contents
MATCH
(p1)-[r1:hasImportantKeyword]->(k:Keyword),
(k)<-[r2:hasImportantKeyword]-(p2:Publication),
(p2)-[:deliverableOf]-(pr1:Project)
WHERE NOT (p1)-[:deliverableOf]-()
WITH
p1.publication_idx AS pub_id, pr1.project_ID AS prj_id,
SUM(r1.score * r2.score) AS score ORDER BY score DESC
WITH
pub_id, collect({pr_id:pr_id, score:score}) AS prediction
RETURN pub_id, prediction[0..5]

-- Query 2
-- Predicting a Publication p1's associated Project
-- using p1's authors' memberships to Projects
MATCH
(p1:Publication)-[:writtenBy]->(u:User),
(u)-[:isMemberOf]->(pr1:Project)
WHERE NOT (p1)-[:deliverableOf]-()
WITH
p1.publication_idx AS pub_id, pr1.project_ID AS prj_id,
COUNT(pr1) AS score ORDER BY score DESC
WITH
pub_id, collect({pr_id:pr_id, score:score}) AS prediction
RETURN pub_id, prediction[0..5]
```

A publication can be related to a project that has other associated publications, based on sharing similar keywords with those other publications. With this intuition, Query 1 (Listing 2) finds potentially correlated projects with a publication. On the other hand, Query 2 (Listing 2) finds projects based on a publication's authors' memberships to projects, as the publication can be a deliverable of those projects. We have combined the two scores for better accuracy of prediction. We discuss the accuracy of link prediction in Section V-E2.

V. Evaluation

We present our analysis, and derive insights on the compute core usage and its impact on the scientific publications (Table II shows the summary). First, we look at statistical inferences from the data sources and then correlate them to derive associative inferences. Our approach has also helped in identifying missing associations based on learning from

the current relationships in the graph. Finally, we present a graphical tool that helps navigate the graph for ascertaining new facts that further complement the graph learning process.

A. Quantitative analysis of Titan's compute core usage

In this section, we present the characteristics of compute core usage of the Titan machine in OLCF from the perspective of users and their science domains. For this analysis, we have studied the constructed graph, for all users who have executed a job on Titan. A vertex of type, *User*, in the graph can have edges of type, *memberOf*, to vertices of type, *Project*, and also map to a vertes of type, *Job*, with an edge of type, *jobExecutedBy*. We have mined for vertices of type, *User*, which have both the edge types. Note that there can also exist vertices of type, *User*, without any edges of type, *memberOf*, in case we cannot find any information from the data sources. We did not consider those vertices in this analysis. We have

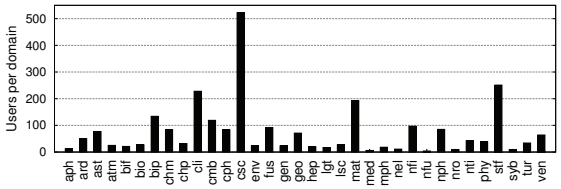
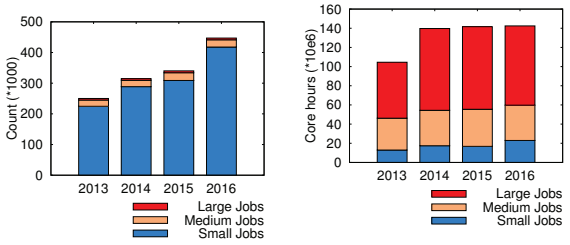


Fig. 3: Number of users per domain.

derived the statistical information about core usage using Cypher queries as described in Section IV-C. Fig. 3 depicts the proportion of users' membership to domains based on edges of type, *memberOf*. The domain *Computer Science, csc,* has the largest number of users (522, 20.50%) and *Staff, stf,* is the second (251, 9.86%). All other domains occupied less than 5% except three domains, *Climate Science, cli,* (228, 8.96%), *Materials Science, mat,* (193, 7.58%) and *Biophysics, bip* (134, 5.26%). Note that though *csc* and *stf* are not science domains, the user base is mostly system engineers and core computer science researchers who directly work on architecting the system and scaling system software on the compute platform. Also, they work closely with domain scientists for scaling application code, and such contributions result in significant publications for core computer science research as well as for the domains. Hence we list both these user communities as science domains in our analysis.

Fig. 4 shows the aggregate number of jobs and compute core hours used over our analysis period of four years. We have categorized the jobs by the number of compute nodes requested, e.g., jobs using less than 375 nodes as small jobs, between 375 and 3750 as medium jobs, and over 3750 nodes as large jobs.

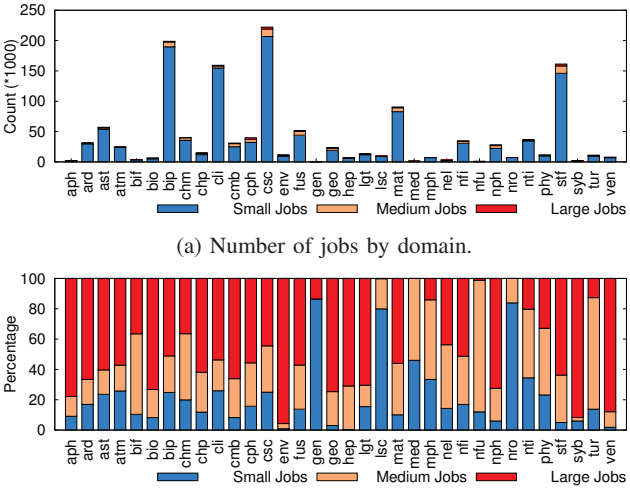
The total number of jobs increased from 2013 to 2016. In terms of used core hours, there is a big increase from 2013 to 2014, but we see similar core hours used from 2014 to 2016. Titan was only commissioned in 2013 and hence the lower utilization. Although the majority of the jobs executed on the system are small-sized jobs, the large jobs consume the



(a) Jobs by the year.

(b) Core hour usage by the year.

Fig. 4: Jobs and core hour usage.



(a) Number of jobs by domain.

(b) Compute core usage by domain (%).

Fig. 5: Compute core usage by domain.

maximum compute core hours on the system, consistent with OLCF's leadership science mission. In terms of job sizes, most small jobs tend to be test or debug runs by users testing newer versions of applications or small scale tests in preparation for a larger run. Alternatively, certain applications or science domains inherently cannot scale beyond a set of nodes.

Next, we have interpreted the compute core usage by science domains. Fig. 5a shows the number of jobs executed by users in the various science domains. As expected, the number of users per domain tends to be correlated with the number of executed jobs, i.e., more users tend to execute more jobs (e.g., *Computer Science, csc*, *Biophysics, bip*, *Climate Science, cli*, *Staff, stf*, *Materials Science, mat*). It is interesting to see that certain domains like *Nuclear Fusion, nfu*, primarily executed medium-sized jobs compared to other domains. Similarly, Fig. 5b presents the compute core usage based on job size. Surprisingly, while certain projects like *Plasma Physics (env), High Energy Physics (hep), Systems Biology (syb)* had a small fraction of job requests, but most of them were large jobs with high core utilization. This analysis served as a clear indicator to identify applications that scaled well on the computing platform and were potential candidates for scaling on future

TABLE II: Summary of users' publication and system usage activities across different science domains. User#(job# > 1) means users who have run more than 1 job; User#(pub# > 1) means users who have more than 1 paper; User#(pub# > 1, job# > 1) means users with more than 1 job and 1 paper. A higher *active user ratio* means more users participated in job executions and/or paper publications.

Science Domain Name (Domain ID)	user#	user# (job#>1)	user# (pub#>1)	user# (#job>1, pub#=0)	user# (pub#>1, job#=0)	user# (pub#>1, job#>1)	user# (pub#=0, job#=0)	Active User Ratio (Job)	Active User Ratio (Publication)	Active User Ratio (Both)	Inactive User Ratio
Accelerator Physics (*aph*)	12	12	6	6	0	6	0	1.000	0.500	0.500	0.000
Aerodynamics (*ard*)	50	43	11	32	0	11	7	0.860	0.220	0.220	0.140
Astrophysics (*ast*)	77	65	28	41	4	24	8	0.844	0.364	0.312	0.104
Atmospheric Science (*atm*)	25	23	8	15	0	8	2	0.920	0.320	0.320	0.080
Bioinformatics (*bif*)	21	18	7	11	0	7	3	0.857	0.333	0.333	0.143
Biology (*bio*)	27	27	16	11	0	16	0	1.000	0.593	0.593	0.000
Biophysics (*bip*)	134	117	49	73	5	44	12	0.873	0.366	0.328	0.090
Chemistry (*chm*)	84	83	45	39	1	44	0	0.988	0.536	0.524	0.000
Physical Chemistry (*chp*)	32	29	16	14	1	15	2	0.906	0.500	0.469	0.063
Climate Science (*cli*)	228	176	67	121	12	55	40	0.772	0.294	0.241	0.175
Combustion (*cmb*)	119	100	28	73	1	27	18	0.840	0.235	0.227	0.151
Condensed Matter Physics (*cph*)	83	78	41	39	2	39	3	0.940	0.494	0.470	0.036
Computer Science (*csc*)	522	437	174	281	18	156	67	0.837	0.333	0.299	0.128
Plasma Physics (*env*)	23	23	17	6	0	17	0	1.000	0.739	0.739	0.000
Fusion Energy (*fus*)	92	80	34	48	2	32	10	0.870	0.370	0.348	0.109
General (*gen*)	23	13	3	10	0	3	10	0.565	0.130	0.130	0.435
Geosciences (*geo*)	71	58	19	40	1	18	12	0.817	0.268	0.254	0.169
High Energy Physics (*hep*)	19	14	12	4	2	10	3	0.737	0.632	0.526	0.158
Lattice Gauge Theory (*lgt*)	17	15	4	11	0	4	2	0.882	0.235	0.235	0.118
Life Sciences (*lsc*)	27	19	4	15	0	4	8	0.704	0.148	0.148	0.296
Materials Science (*mat*)	193	177	78	102	3	75	13	0.917	0.404	0.389	0.067
Medical Science (*med*)	6	6	2	4	0	2	0	1.000	0.333	0.333	0.000
Molecular Physics (*mph*)	18	15	1	14	0	1	3	0.833	0.056	0.056	0.167
Nanoelectronics (*nel*)	11	9	3	6	0	3	2	0.818	0.273	0.273	0.182
Nuclear Fission (*nfi*)	97	88	28	61	1	27	8	0.907	0.289	0.278	0.082
Nuclear Fusion (*nfu*)	4	4	0	4	0	0	0	1.000	0.000	0.000	0.000
Nuclear Physics (*nph*)	85	69	43	35	9	34	7	0.812	0.506	0.400	0.082
Neuroscience (*nro*)	9	8	2	6	0	2	1	0.889	0.222	0.222	0.111
Nanoscience (*nti*)	42	41	19	22	0	19	1	0.976	0.452	0.452	0.024
Physics (*phy*)	38	35	17	18	0	17	3	0.921	0.447	0.447	0.079
Solar/Space Physics (*pss*)	0	0	0	0	0	0	0	N/A	N/A	N/A	N/A
Staff (*stf*)	251	155	69	96	10	59	86	0.618	0.275	0.235	0.343
Systems Biology (*syb*)	8	6	3	4	1	2	1	0.750	0.375	0.250	0.125
Turbulence (*tur*)	34	32	2	30	0	2	2	0.941	0.059	0.059	0.059
Vendor (*ven*)	64	51	0	46	0	5	13	0.797	0.000	0.078	0.203
Average	72.742	60.742	24.457	38.228	2.085	22.514	9.914	0.864	0.332	0.314	0.115

OLCF systems.

B. Quantitative analysis of OLCF acknowledged publication

Publications and citations serve as primary indicators to understand the impact of scientific productivity from leadership computing facilities like OLCF. Fig. 6 presents publications that acknowledged OLCF over the period of four years. As described in Section II, not all publications in a given year were identified and associated with a given project allocation. We have composed graph analytics queries using Cypher as explained in Section IV-C, to extract publication statistics from the constructed graph. In the case of publications in 2015, 91.47% of the publications have their associated project IDs, as opposed to only 50.34% in 2013. In Section IV-E2, we presented how our graph approach can help in identifying this missing information. However, to analyze the publication trends of the different domains, we have only used the publications that have their associated project IDs, and did not use the ones with missing information. Later in the section, we validate and discuss the accuracy of our approach on predicting the missing information.

In Fig. 7, we show the break down of publications and citations by science domains. The *Climate Science (cli)*, *Materials Science (mat)*, and *Nuclear Physics (nph)* domains have the most number of publications (44.71% of total) and citations (56.12% of total). The number of citations per domain is highly correlated with the number of publications per domain, based on the Pearson Correlation Coefficient (PCC) that was

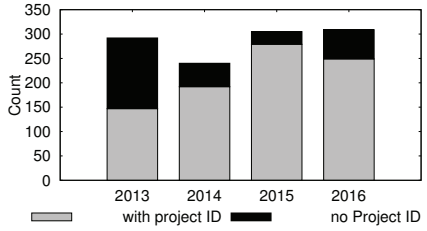
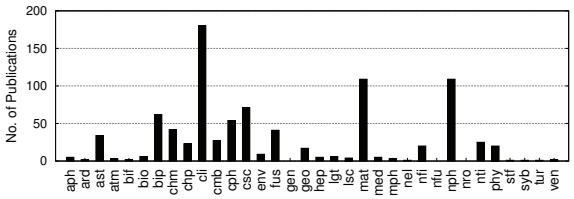


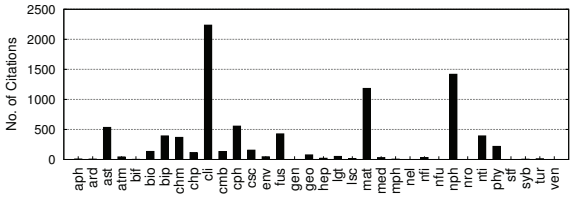
Fig. 6: Publication count per year.

0.95. PCC computes the linear correlation score between two variables X and Y as follows: $\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}$, cov is the covariance, σ_X and σ_Y are the standard deviations of X and Y, respectively. However, some domains such as *Biology (bio)*, *Physical Chemistry (cph)*, *Climate Science (cli)* and *Astrophysics (ast)* have high citation counts but with less number of publications. To study the influence of individual papers in different domains, we have calculated the number of citations divided by the number of publications. Fig. 8 shows that publications from the *Biology (bio)* domain has the highest ratio (#citations/#publications = 22.33).

Using the author's affiliation with a science domain, we have used the publication data to understand collaboration trends in inter/intra science domains. First, we have compared the average number of authors per publication in different domains, which is representative of the collaborating team size. Each publication's science domain is identified from the



(a) Publication count per domain.



(b) Citation count per domain.

Fig. 7: Publication and citation per domain.

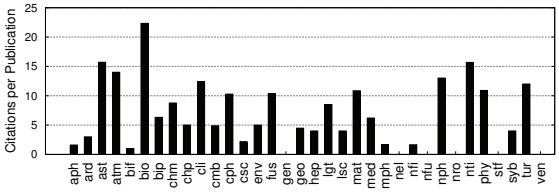


Fig. 8: Citations per publication.

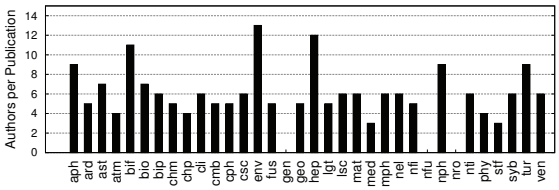
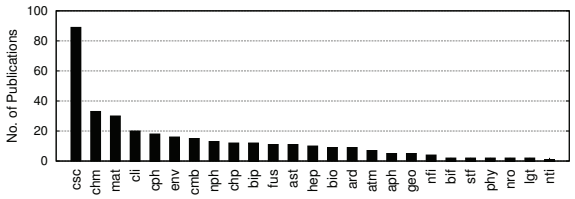


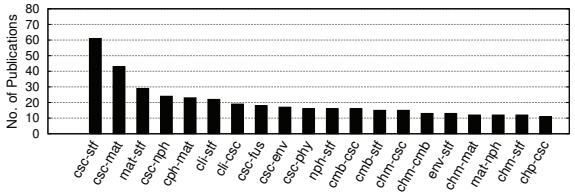
Fig. 9: Average number of authors per publication.

publication's associated project. Fig. 9 shows that, on average, each publication has 6.10 authors. The *Plasma Physics (env)* domain, on an average, had the most number of authors (13.22) per publication.

Next, we have analyzed how frequently each science domain collaborates with other domains. To this end, for each domain X, we counted the number of publications having an author who is a member of the domain X and at least one another who is a member of another domain that is not X. Fig. 10a shows that *Computer Science (csc)* collaborates with the other domains the most. It is a reasonable result, because in many cases, computer scientists take a role for supporting scientists from other scientific domains to be able to leverage OLCF facilities for their research. *Chemistry (chm)* and *Materials Science (mat)* are the next two that collaborate the most with other domains. Fig. 10b shows the pairs of science domains that tend to collaborate more. Considering that *Staff (stf)* is a generic domain, we can see that scientists of *Computer Science (csc)-Materials Science (mat)*, *Computer Science (csc)-Nuclear Physics (nph)*, *Condensed Matter Physics (cph)-Materials Science (mat)* domains tend to collaborate more with each other than the other pairs of domains.



(a) Collaboration frequency, number of publications with authors from more than one domain.



(b) Collaboration frequency of two domains.

Fig. 10: Collaboration.

C. Correlation analysis between Titan usage and publications

We have analyzed the correlation between publications and compute core usage. As explained earlier, statistical information about publication and compute core usage are achieved from the graph. Thereafter, we have performed additional computations externally to derive correlations. We observe that users in general launch more small jobs, although the primary compute usage is only because of large jobs. To understand how the job size relates to publication, we have calculated the Pearson correlation coefficient between the compute hours used for different job size bins and the count of publications for individual science domains (Fig. 12).

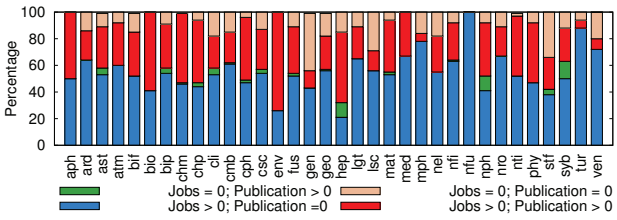


Fig. 11: (%) of User Contribution.

As seen in Fig. 12, core hours used is highly correlated to both the number of publications and citations compared to the number of users in domains. Domains that use more core hours tend to have more publications and citations. Interestingly, core hours consumed by small jobs is highly correlated with both publication and citation counts, because scientific conduct leading up to publications involves running many small test and development jobs.

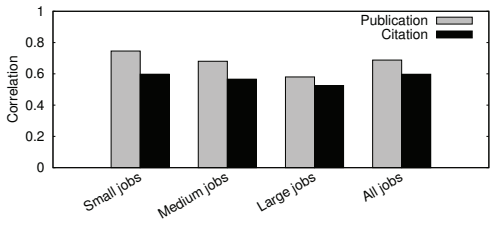


Fig. 12: Publication and Citation correlation to the job size(compute core hours).

Not all job executions lead to publications. We have analyzed the ratio of job executions and publications to see how it varies across different domains. Specifically, for each science domain, we have computed the normalized publication count divided by the normalized total core hours used by the domain. As depicted in Fig. 13, *Climate Science (cli)* shows the highest number, followed by *Nanoscience (nti)* and *Molecular Physics (mph)*.

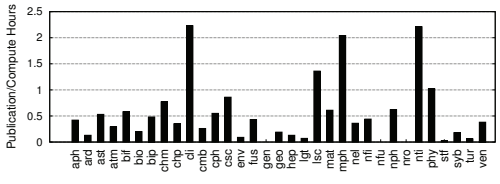


Fig. 13: Publication per compute core hour (The domain *med* has been excluded in this figure as its value 57.810 was an outlier. The total core hour used by the domain was insignificant and the publications resulted from compute used earlier than our analysis period.)

D. Text analysis of publications for OLCF research trends

In addition to the standard entities (users, jobs, publications and projects) in our graph, we have also introduced keywords, extracted from the text of publications, as vertices. This helps us in understanding research trends and similarity between science domains based on keywords appearing in publications. A key outcome of this analysis is the ability to summarize the top keywords that describe research trends in publications for each year. We have realized this in the graph in the following fashion. Recall that the graph has keywords connected to publications using the edge type, *hasImportantKeyword*, with a property, *score* that depicts the strength of the edge. To compute the ranking of a given keyword, we have summed the *score* values of the edges of type, *hasImportantKeyword*. The Cypher query below (Listing 3) shows how we computed the top 10 keyword trends in this manner, for the year 2013 (see Table III for ranking).

Listing 3: Finding top-10 publication trend keywords.

```
MATCH (k:Keyword)<-[r:hasImportantKeyword]-(p:Publication)
WHERE p.year=2013
RETURN k.keyword_term,sum(r.score) as ranking_score
ORDER BY ranking_score desc
LIMIT 10
```

We have found that the keyword *nuclear* occurred consistently in all four years. Several keywords appeared multiple times such as *ice*, *fission*, *magnetic*, *precipitation*, *water*, *io*, *graphene* and *performance*.

TABLE III: Top 10 publication trend keywords.

Rank	2013	2014	2015	2016
1	nuclear	nuclear	staff	nuclear
2	reconnection	plasma	flame	water
3	ice	graphene	combustion	atlas
4	performance	ice	quantum	polymer
5	transport	precipitation	nuclear	fission
6	turbulence	instability	openshmem	detector
7	electron	ionization	ice	qcd
8	nuclei	land	pi	tev
9	magnetic	io	magnetic	particle
10	graphene	water	variability	pi

The previous discussion highlighted how we derived top 10 keywords representative of the publications in a given year. Next, we discuss the derivation of the top 10 keywords that are representative of the jobs that were run on OLCF. To address this, we need the keyword to job score. However, since the jobs logs are devoid of any keywords, semantically correlating keywords to job execution trends is not a straightforward task. Instead, we need to obtain a score for the graph pattern, $\{$*keyword, publication, job*$\}$. Note that we already have the connection between publication and keyword, and a relevance score for the same. To derive the job to publication connection, we have made the following inference: a publication is related to a job if it was authored by and executed by the same user, for the same project and that the publication occurred during a certain *time window*. We have assumed a period of 1 to 2 years after job execution for the time period, which is how long users typically take to publish after job runs. The longer the core hour usage of a job, the higher the score of the link between the job and the publication. We now have two scores, which we have averaged to derive the score for the graph pattern, $\{$*keyword, publication, job*$\}$. The Cypher query below (Listing 4) shows how we computed the top 10 keywords representing the jobs run at OLCF for the years 2013-2015 (see Table IV for ranking). Note that we have filtered out generic terms (e.g. "opinion," "diverse," etc.) from the list, and adjusted the ranking.

Listing 4: Computing base scores for ranking job trend keywords (2013).

```
MATCH (j:Job)-[:jobExecutedBy]->(u:User),
(u)<-[:writtenBy]-(p:Publication),
(p)-[:deliverableOf]-(pr:Project),
(j)-[:jobExecutedFor]-(pr),
(k:Keyword)<-[r:hasImportantKeyword]-(p:Publication)
WHERE j.year=2013 and j.year+1<=p.year and p.year<=j.year+2
RETURN k.keyword_term, sum(r.score) as keyword_score,
sum(j.num_of_nodes*j.execution_time) as core_hour_score
```

In a similar fashion, we can compute the similarity between science domains based on common keywords, which could help foster future collaborations.

TABLE IV: Top 10 job trend keywords.

Rank	2013	2014	2015
1	enzymatic	lattice	transporters
2	biomass	resonance	scattering
3	molecular	molecular	ions
4	core-collapse	pi	allosteric
5	magnetic	qcd	bar
6	supernovae	spectrum	interfacial
7	inhibition	scattering	sodium-coupled
8	lignin	cosmological	qcd
9	dot	bar	fusion
10	neutrino	retroviral	spectrum

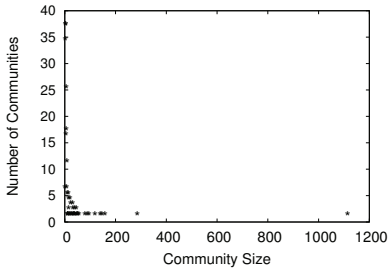


Fig. 14: Distribution of community size based on the vertex type, *User*.

E. Utilization of graph mining for association analysis

1) Community detection: As a use case of community detection described in Section IV-D1, we have detected communities with vertices of type, *User*, in the constructed graph, based on their co-authorship connections. We have only considered *User* vertices with at least one publication. The total number of identified communities was 246. The distribution of community sizes follows the power-law distribution with a long-tail, as there are a few large communities and a large number of small communities (Fig. 14). The average size of a community was 17.30. The top 10 large communities cover more than 53.69% of the users that were considered in this analysis, which means that the majority of the users are in a large collaboration network. The largest community had 1,115 users, and the number of publications by the largest community was 345, which is a very large portion (30.10%) of the total number of publications. To see which domains are part of such a large community, we have analyzed the membership of users in the community to science domains. We observe that a large portion of users belonged to *Climate Science (cli)*, *Computer Science (csc)*, *Physical Chemistry (cph)*, and *Biophysics (bip)*, and their publications are highly associated with terms such as *ice*, *data*, *io*, *climate*, etc. The most common sizes of communities were 3 and 4 (37 communities each). This result can be useful for OLCF to identify small collaboration networks, and connect them with larger communities that are working on similar research topics, which can be identified by the keywords associated with the communities.

2) Accuracy of link prediction via relationship strength quantification: One of the objectives of our analysis was to enable the association of scientific publications to OLCF project allocations. As described earlier, associating the publications to projects is currently a tedious and manual process. Our graph approach has helped predict missing project ID for publications, and we have further verified the accuracy of the prediction to validate the approach. As described in section IV-D2, we have composed two different queries that perform the quantification of relationship strength, one based on the contents of publications (Query 1), and the other based on the authors' memberships to projects (Query 2). Each query produced a set of results, (project ID, score), ordered by the score in descending order. For each project ID, we have also computed a weighted sum of two scores generated by query 1 and query 2, and used the value as a hybrid ranking score for the project ID. w_1 and w_2 are weights assigned to the scores of query 1 and query 2, respectively. If $w_1 = 1.0$ and $w_2 = 0.0$, only query 1 was used, and if $w_1 = 0.0$ and $w_2 = 1.0$, then only query 2 was used.

For the evaluation, we have used the publications that already have the associated project ID information to create an answer set for the prediction test. We pretend that these relationships do not exist in the graph, and tested if the prediction technique can identify these relationships. For each test case, we have estimated the project ID for the given publication, p, and created a ranked list of project IDs by sorting them in descending order using the score. Thereafter, we measured the Top-k performance of the ranked list using recall(k) [10] as follows. Recall(k) represents how accurately our prediction can perform when generating a top-k ranked list. Let r be the rank of the project ID, p. If $r \leq k$, we have a hit, otherwise, a miss. Then, recall(k) = $\frac{\#hits}{|T|}$ where $|T|$ is the number of tests.

Fig. 15 shows that query 1 (w_1=1.0, w_2=0.0) outperformed query 2 (w_1=0.0, w_2=1.0). Hybrid usage of these two queries (w_1=0.5, w_2=0.5) showed the best performance, where recall(k) is 0.78 and 0.86 for k=1 and k=5, respectively. The result shows that using various paths between vertices can improve the accuracy of predicting missing links.

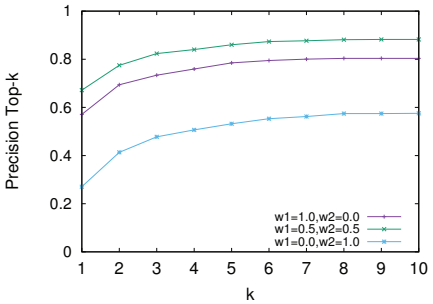


Fig. 15: Prediction accuracy of unknown project IDs.

VI. OLCF Graph Network Analytic Tool

To put things together, we have a built web-based graphical user interface analytic tool atop the constructed graph, the queries, operations, and database that we have presented in this paper. For the implementation, we have utilized vis.js [11], Bootstrap, jQuery [12] and its DataTables plugin for the graph visualization and web-interface, Tornado webserver [13] and Py2neo python neo4j API for the server, and neo4j for the back-end graph database. The tool was used to visualize the graph and the analytics query results presented in this paper. Although predefined analytics queries provide various insights, the flexibility of searching and navigating through the relationships can be very useful for individual user's specific needs. To this end, the tool allows users to search vertices in the graph by keywords, navigate the search results by following the edges, finding missing links between projects and publications. It also has a panel where users can set up constraints to filter out vertices and edges. Fig. 16 presents screenshots of the dashboards provided by the developed tool.

VII. Related Work

Constructing a graph from various data sources is driven by the data model and query needs. Lee et al. [7] showed how to create a heterogeneous graph from large-scale tabular data sources via mapping between relational and graph schemas. We have extended this approach by incorporating LSA [4] to capture semantic relevance scores between documents and keywords.

Our graph analysis was focused on acquiring statistical characteristics of the graph based on various subgraph pattern matching referred to as *Online Graph Analytic Processing (OLGAP)* [14]. There are a number of graph engines that can support OLGAP analysis such as neo4j [2], DEX [15], Sesame [16], etc. The graph engine Sesame [16], using RDF (Resource Description Framework) as its data model and SPARQL [17] for the query language, is a widely used graph data model and query language, due to its simplicity and flexibility. However, they lack the definition of types and attributes for nodes and edges that are crucial for analysis of complex real-world data sets. We chose neo4j [2] and its query language *Cypher* [3] due to its underlying property graph-based data model that can represent complex node/edge types and their attributes more naturally. *Cypher* is an SQL-inspired declarative language for describing patterns, started as a part of a commercial software, now being standardized by the *openCypher* project.

Another important aspect of graph analysis is *Graph Mining (GM)*, which focuses on the automatic discovery or prediction of graph properties (e.g., counting triangles [18], finding eccentricity [19], finding connected components, computing PageRank/Personalized PageRank [20], [21]) based on pre-defined patterns. We have applied these techniques in our work. In [10], the authors show that scoring relationship strength between nodes using various paths can be used for recommender systems. In a similar vein, we have used various graph pattern queries to quantify the score for link prediction between vertices. As future work, we could potentially leverage other graph mining approaches such as PageRank for the identification of important vertices in the network.

In the HPC domain, the increased usage of GPUs has driven the need for scaling graph libraries on compute platforms. However, there has been very limited work on using graph analytics to manage and understand such large-scale systems. Constellation [22], was aimed at defining a science graph network for fostering collaboration and knowledge discovery across science domains, exploring similarities in the graph structures across domains. In [23], the authors used semantic graph techniques for cluster management, associating the job scheduler's log to failure events and looking for patterns of failure. Similarly, [24] describes a distributed graph processing approach for analyzing HPC system logs in real-time. In this paper, we have used graph analytics for correlation analysis between system usage and publications, and to derive new edge mappings based on observed graph patterns.

VIII. Conclusions

In this paper, we have showcased a novel application of graph-based big data techniques to glean insights on the OLCF HPC center's compute core usage and publication trends. To this end, we have represented the OLCF HPC resource fabric as a graph; the entities within the fabric such as users, scientific projects, groups, jobs and publications as vertices in the graph; the relationships between the entities as edges, connecting the vertices; and, applied graph analytics, text analysis and graph mining to understand heterogeneous data sets for a leadership-scale HPC center. Our results provide valuable insights into an HPC center's core allocation program, measuring the productivity of scientific domains, the interplay between core usage and research output, semantic overviews regarding the contents (publications) of the OLCF scientific projects from various perspectives, accelerating collaboration, entity relationship analysis, and in predicting new connections between HPC resource entities.

References

[1] "OLCF." [Online]. Available: https://www.olcf.ornl.gov/

[2] J. J. Miller, "Graph database applications and concepts with neo4j," in *Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA*, vol. 2324, 2013.

[3] J. Marton, G. Szárnyas, and D. Varró, "Formalising opencypher graph queries in relational algebra," *arXiv preprint arXiv:1705.02844*, 2017.

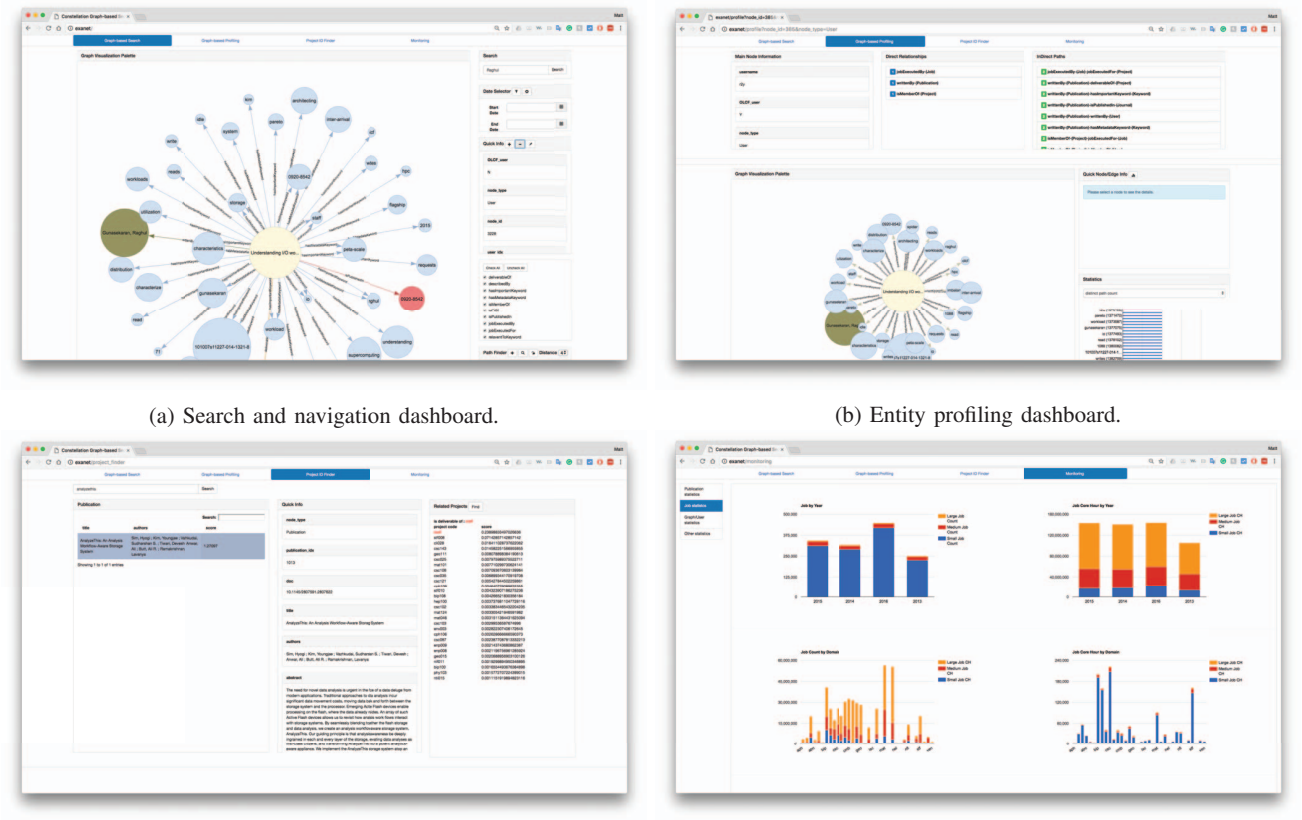
[4] T. K. Landauer, P. W. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse processes*, vol. 25, no. 2-3, pp. 259–284, 1998.

[5] H. S. Kunii, *Graph Data Model: And Its Data Language*. Springer Science & Business Media, 2012.

[6] M. A. Rodriguez and P. Neubauer, "Constructions from dots and lines," *Bulletin of the American Society for Information Science and Technology*, vol. 36, no. 6, pp. 35–41, 2010.

[7] S. Lee, B. H. Park, S.-H. Lim, and M. Shankar, "Table2graph: A scalable graph construction from relational tables using map-reduce," in *Proceeding of the 1st IEEE BigDataService*. IEEE, 2015, pp. 294–301.

[8] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the 1st instructional conference on machine learning*, 2003.



(a) Search and navigation dashboard.

(b) Entity profiling dashboard.

(c) Missing link prediction dashboard.

(d) OLCF core usage and publications data monitoring dashboard.

Fig. 16: OLCF graph network analytic tool.

[9] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische mathematik*, vol. 14, no. 5, pp. 403–420, 1970.

[10] S. Lee, S. Park, M. Kahng, and S.-G. Lee, "Pathrank: Ranking nodes on a heterogeneous graph for flexible hybrid recommender systems," *Expert Systems with Applications*, vol. 40, no. 2, pp. 684–697, 2013.

[11] B. Almende, "vis. js–a dynamic, browser based visualization library," *URL http://visjs. org/. Acesso em*, vol. 1, 2016.

[12] E. McCormick and K. De Volder, "Jquery: finding your way through tangled code," in *19th annual ACM SIGPLAN conference*. ACM, 2004.

[13] M. Dory, A. Parrish, and B. Berg, *Introduction to Tornado: Modern Web Applications with Python*. " O'Reilly Media, Inc.", 2012.

[14] S. Lee, S. R. Sukumar, S. Hong, and S.-H. Lim, "Enabling graph mining in rdf triplestores using sparql for holistic in-situ graph analysis," *Expert Systems with Applications*, vol. 48, pp. 9–25, 2016.

[15] N. Martínez-Bazan, V. Muntés-Mulero, S. Gómez-Villamor, J. Nin, M.-A. Sánchez-Martínez, and J.-L. Larriba-Pey, "Dex: high-performance exploration on large graphs for information retrieval," in *Proceedings of the 16th ACM CIKM*. ACM, 2007, pp. 573–582.

[16] J. Broekstra, A. Kampman, and F. Van Harmelen, "Sesame: A generic architecture for storing and querying rdf and rdf schema," in *ISWC 2002*. Springer, 2002, pp. 54–68.

[17] E. PrudHommeaux, A. Seaborne *et al.*, "Sparql query language for rdf," *W3C recommendation*, vol. 15, 2008.

[18] C. E. Tsourakakis, "Fast counting of triangles in large real networks without counting: Algorithms and laws," in *Proceeding of the 8th IEEE ICDM*. IEEE, 2008, pp. 608–617.

[19] P. Hage and F. Harary, "Eccentricity and centrality in networks," *Social networks*, vol. 17, no. 1, pp. 57–63, 1995.

[20] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[21] H. Tong, C. Faloutsos, and J.-Y. Pan, "Fast random walk with restart and its applications," 2006.

[22] S. S. Vazhkudai, J. Harney, R. Gunasekaran, D. Stansberry, S. H. Lim, T. Barron, A. Nash, and A. Ramanathan, "Constellation: A science graph network for scalable data and knowledge discovery in extreme-scale scientific collaborations," in *2016 IEEE International Conference on Big Data (Big Data)*, Dec 2016, pp. 3052–3061.

[23] J. Brandt, V. D. Sapio, A. Gentile, P. Kegelmeyer, J. Mayo, P. ebay, D. Roe, D. Thompson, and M. Wong, "A framework for graph-based synthesis,analysis, and visualization of hpc cluster job data," Sandia National Laboratories, Albuquerque, New Mexico (United States)., Tech. Rep., 2010.

[24] S. Weigert, M. Hiltunen, and C. Fetzer, "Mining large distributed log data in near real time," in *Managing Large-scale Systems via the Analysis of System Logs and the Application of Machine Learning Techniques*, ser. SLAML '11. ACM, 2011.