RC25164 (W1105-074) May 24, 2011
Computer Science

IBM Research Report

Analysis and Modeling of Social Influence in High Performance Computing Workloads

Shuai Zheng[1], Zon-Yin Shae[2], Xiangliang Zhang[1], Hani Jamjoom[2], Liana Fong[2]

[1]King Abdullah University of Science and Technology
Thuwal, Saudi Arabia

[2]IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598 USA

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Analysis and Modeling of Social Influence in High Performance Computing Workloads

Shuai Zheng[‡], Zon-Yin Shae[*], Xiangliang Zhang[‡],
Hani Jamjoom[*], and Liana Fong[*]

[*]IBM T. J. Watson Research Center, Hawthorne, NY
[‡]King Abdullah University of Science and Technology, Thuwal, Saudi Arabia

Abstract. Social influence between users (e.g., collaborating on a project) creates bursty behavior in the underlying high performance computing (HPC) workloads. Using representative HPC and cluster workloads, this paper identifies, analyzes, and quantifies the level of social influence across HPC users. We show the existence of a social graph that is characterized by a pattern of dominant users and followers. This pattern also follows a power-law distribution, which is consistent with those observed in mainstream social networks. Given its potential impact on HPC workload prediction and scheduling, we propose a fast-converging, computationally-efficient online learning algorithm for identifying social groups. Extensive evaluation show that our online algorithm can identify stable social groups after observing only 1% of workload arrivals.

1 Introduction

Wide-use and expansion of collaboration technologies (e.g., social networking) is influencing user behavior across all aspects of his/her day-to-day activities. Almost completely overlooked, this paper analyzes the effects of *social influence* on high-performance computing (HPC) workloads. The intuition is that user collaboration affects the underlying job submission characteristics. For example, students in a class will likely exhibit correlated workload characteristics, especially considering project deadlines, homework, etc.

Discovering the underlying social patterns and dependencies within groups of correlated users—or *communities*, for short—will help improve workload prediction and job scheduling. Our work is akin to those in *community centric web search*, and more recently, to Lin *et al.* [7] which discovers the communities based on mutual awareness from observable blogger actions. Unlike existing studies, this paper—to the best of our knowledge—is the first attempt to propose a social-influence-aware method for discovering correlated users and modeling their corresponding workload in HPC environments.

In an HPC environment, community discovery has several challenges. First, not all the users are regular users of HPC/clusters. Ephemeral users need to be identified and discarded. Second, computing similarities between users is difficult. Since each user submits a different number of jobs to HPC/clusters, measuring the pairwise similarity of users based on their submitted jobs is both complex

and unreliable, especially when the jobs are described by a complex structured language, e.g., Job Description Language (JDL) [10]. Finally, the community discovery process must be computationally efficient—especially for large-scale workloads—so that it can be used to improve the underlying job scheduling.

The challenges outlined above limit the applicability of standard clustering techniques (e.g., double-clustering approach in [12]). In this paper, an efficient method is proposed to identify the *correlated users* from HPC/Cluster workload. Depending on a user's activeness, we characterize his/her as either a *dominant user* or a *follower* to a dominant user.

Following similar analysis of social networks [4], we show that our discovered communities also exhibit power-law characteristics. This has profound implications on the importance of accounting for dominant users in job scheduling. Basically, identifying dominant users and their followers allows job schedulers to better predict change in future resource demands. To enable effective usage of this new insight, we propose an online learning algorithm that dynamically learns the social characteristics of a given workload. Experimental results show that our online algorithm can efficiently identify stable social groups by observing only a small portion—about 1%—of workload arrivals.

The remainder of this paper is organized as follows. Section 2 introduces the data sources we used. We describe our proposed method in Section 3. We then characterize various aspects of the discovered communities in Section 4. Section 5 describes the online learning mechanism. We discuss the related work in Section 6 and conclude the paper in Section 7.

2 Data Sources

The first dataset we used is the Grid 5000 traces [2]—a popular HPC workload testbed. Grid 5000 is an experimental grid platform consisting of nine geographically distributed sites across France. Each site comprises of one or more clusters, for a total of 15 clusters. We use the traces recorded by the individual Grid 5000 clusters from the beginning of the Grid 5000 project (during the first half of 2005) to November 10^{th}, 2006.

While they are many useful parameters for each job record in the trace: we extract only `UserID`, `GroupID`, and `SubmittedTime` for our study here. Every group has thousands of jobs, and the groups are classified by the site location. In the Grid 5000 trace, there are a total of 10 groups, more than 600 users, and more than 100,000 jobs.

The second dataset that we used is the job trace (i.e., the Logging and Bookkeeping (L&B) files) from the Enabling Grid for EScienc E[1] (EGEE) grid. EGEE currently supports up to 300,000 jobs per day on a 24x7 basis. Similar to Grid 5000, we extract the submission `timestamp` and `userID` as job parameters. We use two sets of EGEE L&B files. One contains 229,340 jobs submitted by 53 users in 2005. The other has 347,775 jobs submitted by 74 users in 2007.

[1] http:// www.eu-egee.org/

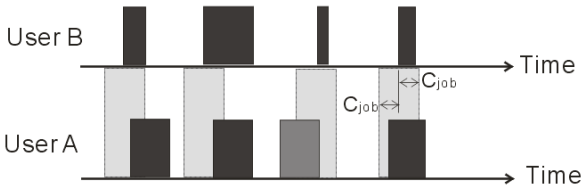

Fig. 1. An example of socially influenced jobs. User B has $3/4 = 75\%$ of jobs that can find a submission of A within C_{job} before/after their submission time.

3 Social Influence Model

In this section, we define *social influence matrix* between users. We focus on studying social relationships based on the submission time of jobs. As an example, consider two users working on a project, which consists of many of jobs. If the two users are working closely on the project (e.g., paper deadline), their job submission times will be close. We refer to the two users and their jobs as being *socially influenced*. In this paper, we use two key features, `UserID` and `SubmittedTime`, to analyze the social influence between users. We do not consider the duration time of jobs.

Take Group 1 in the Grid 5000 dataset as an example. In total, we have 38 different users, each submitting hundreds of jobs. Furthermore, consider two users: User A (submitted 1000 jobs) and User B (submitted 800 jobs). Social influence between these two users are captured by two factors:

- **Socially-connected jobs**: for a job of User B, if the minimum time between the submission time of this job and at least one of the 1000 jobs of User A is small enough (e.g., less than half-hour before/after submission time), then we say this job is socially connected to User A. We refer to this minimum time threshold between jobs as C_{job}.
- **Socially-connected users**: for the entire 800 jobs of User B, if more than $x\%$ (e.g., $x = 50\%$ or 80%) of the jobs are socially connected jobs to User A, then we say User B is socially connected to User A. We refer to the $x\%$ threshold as C_{user}. Furthermore, we define User A as a **dominant user**, and User B as a **follower** of User A.

It should be noted that the social connections are directed relationships. The fact that User A is socially connected to User B does not mean that User B is also socially connected to User A.

The social influence between User A and User B are depicted in Figure 1. In this example, three out of User B's four jobs can find at least one submission from User A within the time interval C_{job}. We can say that User B has 75% of

Algorithm 1 Algorithm of Calculating the Social Influence Matrix

Input: Data set of jobs $D = \{UserID,\ JobTime\}$,
including the $UserID$ and *submitted time* of a job
$U = unique(UserID)$
Criterion $C_{job} = 0.5\ hour, 1\ hour, 6\ hours$
Criterion $C_{user} = 50\%, 80\%$
Output: Social Influence Matrix M
for j = 1 to $|U|$ **do**
$Y = \{JobTime(q)|UserID(q) = U_j\}$
(all jobs submitted by U_j)
for i = 1 to $|U|$ **do**
$X = \{JobTime(q)|UserID(q) = U_i\}$
(all jobs submitted by U_i)
for k = 1 to $|X|$ **do**
$d(k) = min(|X(k) - Y(q)|),\ q = 1, ..., |Y|$
end for
$M(i,j) = \sum_{k=1}^{|X|} (d(k) < C_{job}) / |X|$
if $M(i,j) > C_{user}$ **then**
user U_i is socially connected to user U_j, and
user U_i is a follower to the dominant user U_j
end if
end for
end for

jobs socially connected with User A, and he/she will be a follower to User A if C_{user} is set to a value less than 75%.

Next, we turn our attention to building the *social influence matrix*. For ease of processing, we sort the 38 users in Group 1 of the Grid 5000 according to their `UserID`. The social influence matrix is a 38 by 38 matrix, noted as M. The element $M(i,j)$ of i-th row and j-th column denotes the corresponding percentage of the jobs of user i that are socially connected to user j. We propose Algorithm 1 to calculate the social influence matrix.

Table 1 shows the *social influence matrix* of Group 1 when C_{job} is one hour. We give influence matrix of the first five users. In the first column, the first element is 1, which means that 100% of User 1's jobs are socially connected to each other. Obviously, all diagonal values of the matrix are 1. The second element in the first column is 0, which means that none of the jobs of second user are socially connected to the first user.

As described earlier, social connectedness between users is influenced by the threshold value C_{user}—the minimum percentage of socially-connected jobs. If we set the criterion C_{user} to 80%, the *followers* to User i are the ones who have values larger than 0.8 in i-th column. We show the number of *followers* in Table 2 for C_{user}=80% and C_{user}=50%. As expected, as C_{user} is decreased, we have an increase in the number of followers. For example, User 2 has 5 followers when C_{user} is reduced to 50% from 80%.

Table 1. One-Hour Social Influence Matrix

	User 1	User 2	User 3	User 4	User 5	...
User 1	1	0	0	0	0	...
User 2	0	1	0	0.029	0.084	...
User 3	0	0	1	0	0	...
User 4	0	0.131	0	1	0.239	...
User 5	0	0.048	0	0.056	1	...
...	...	...	...	...	...	...

Table 2. Number of Followers. A value of 1 means that there is only one follower to the corresponding user, or this user is only socially connected to himself/herself.

C_{job} C_{user}	User 1	User 2	User 3	User 4	User 5	...
1 hr 80%	1	3	1	2	1	...
1 hr 50%	1	5	1	2	2	...

4 Analysis of Social Influence

In this section, we analyze the Grid 5000 and the EGEE data to discover socially-connected users. We are interested in studying the impact of the values for C_{job} and C_{user} on social connectivity. Figure 2 is the Cumulative Distribution Function (CDF) distribution of the jobs' interarrival time for the four datasets. We find that all four datasets have more than 95% of their jobs' interarrival times smaller than 0.5 hour. So, we choose $C_{job} = 0.5$ *hour* in following sections. The criterion C_{job} is set to 6 hours, 1 hour, or 0.5 hour, while the criterion C_{user} is set to 50% or 80%.

4.1 Community Extraction from HPC Workloads

Figure 3 (a) shows the socially connected users (number of followers) for each user identified from Group 1 of Grid 5000 trace with criterion $C_{user} = 50\%$. When $C_{user} = 50\%$ and $C_{job} = 6$ *hours*, User 1 has 13 followers, User 2 has 11 followers, and User 3 has 10 followers. When $C_{user} = 50\%$ and $C_{job} = 1$ *hour* or $C_{job} = 0.5$ *hour*, the number of followers for every user decreases as expected. In all figures of this paper, the users are ordered by the number of followers with $C_{user} = 50\%$ and $C_{job} = 6$ *hours*.

Figure 3 (b) shows the number of followers for each user in Group 1 with criterion $C_{user} = 80\%$. When $C_{user} = 80\%$ and $C_{job} = 6$ *hours*, User 5 has 6 followers, User 1, User 2, User 3 and User 4 have 5 followers. When $C_{user} = 80\%$ and $C_{job} = 1$ *hour* or $C_{job} = 0.5$ *hour*, the number of followers for every user decreases. For comparison, Figure 3 (c) and Figure 3 (d) show the number of followers for each user in Group 6 with different settings of C_{user} and C_{job}. Similarly, the Figure 4 shows the number of followers distribution identified from workloads of EGEE 2005 and EGEE 2007.

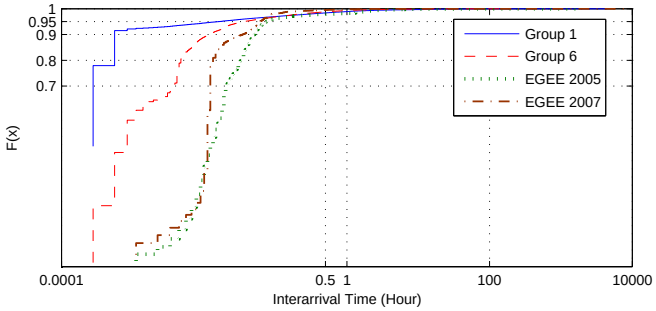

Fig. 2. CDF of jobs' interarrival time for Group 1, Group 6, EGEE 2005 and EGEE 2007

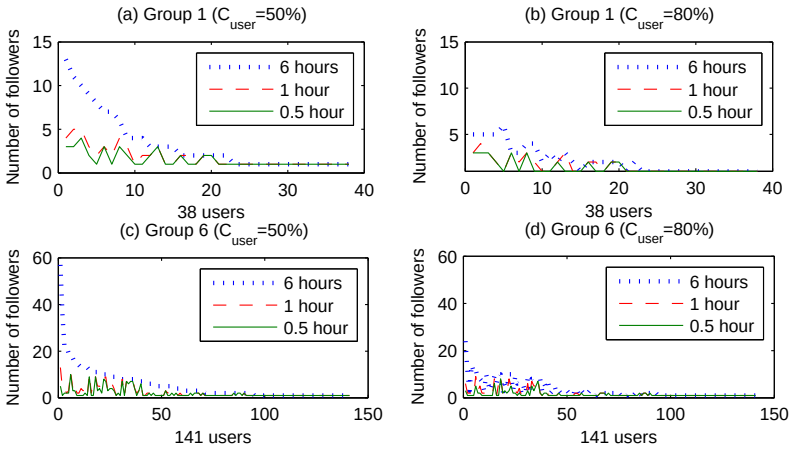

Fig. 3. Social groups discovered in Group 1 and Group 6 of Grid 5000 traces showing the number of followers to each dominant user with different criteria of C_{user} = 50%, 80% and C_{job} = 6 *hours*, 1 *hour*, 0.5 *hour*

4.2 Power-law Distribution of Discovered Communities

A common property of many large networks is that the vertex connectivities follow a scale-free power-law distribution [4]. We would like to investigate if such a power-law property exists among HPC users. In our study, each vertex is a person. The connectivity between vertices can be used to describe if two users interact with each other.

Let $P(k)$ denote the probability that a user has a number of followers k, where k is a positive integrate number. In order to show our discovered social group have the same property as the common networks, we investigate whether the number of followers k of each dominant user has the power-law distribution, $P(k)$=$a*k^b$. Figure 5 shows the power-law distribution of the number of followers identified

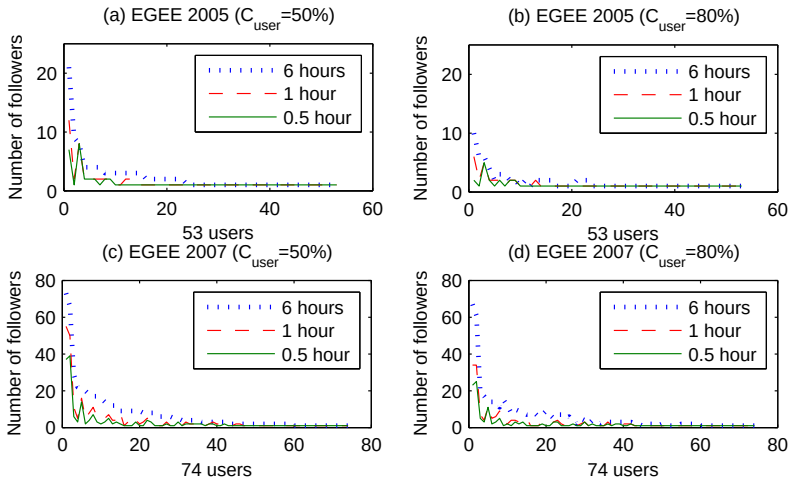

Fig. 4. Social groups discovered in EGEE 2005 and EGEE 2007 traces showing the number of followers to each dominant user with different criteria of $C_{user} = 50\%$, 80% and $C_{job} = 6\ hours$, 1 *hour* and 0.5 *hour*

from Group 1 and 6 of Grid 5000, as well as from EGEE 2005 and EGEE 2007. The power-law distribution of other groups exhibit similar characteristics (unless the group size was very small); they are not shown for space considerations. All social followers are discovered with connected job criterion $C_{job} = 0.5\ hour$ and connected user criterion $C_{user} = 50\%$. From Figure 5 we can see that the number of followers fits very well the power-law distribution with different parameter a and b.

5 Design of Online Learning Mechanism

Our earlier analysis used an offline mechanism to identify social influence in HPC workloads. For our analysis to be consumable by HPC job schedulers and resource managers, a real-time (online) mechanism is needed. Algorithm 2 shows the proposed mechanism for computing the social influence matrix on the fly while jobs are arriving. Suppose that at time $JobTime^t$ a user $UserID$ submits a job, we have $(UserID^t, JobTime^t)$ as a sample in our streaming workload, where time step $t \geq 1$. Given the threshold C_{job} of socially connected jobs, we maintain a $Un^t \times Un^t$ matrix M^t, where Un^t is the number of unique users until t. Each element of M^t is a 2-tuple object $M_{ij}^t = \{R_{ij}^t, C_i^t\}$, where C_i^t is the number of jobs submitted by user $UserID = U(i)$ until time t, and R_{ij}^t is the number of jobs of $UserID = U(i)$ that are socially connected to $UserID = U(j)$ until t.

We use MATLAB to implement and simulate Algorithm 2. Unlike Algorithm 1, we use a sliding time window along the workload flow. This window only

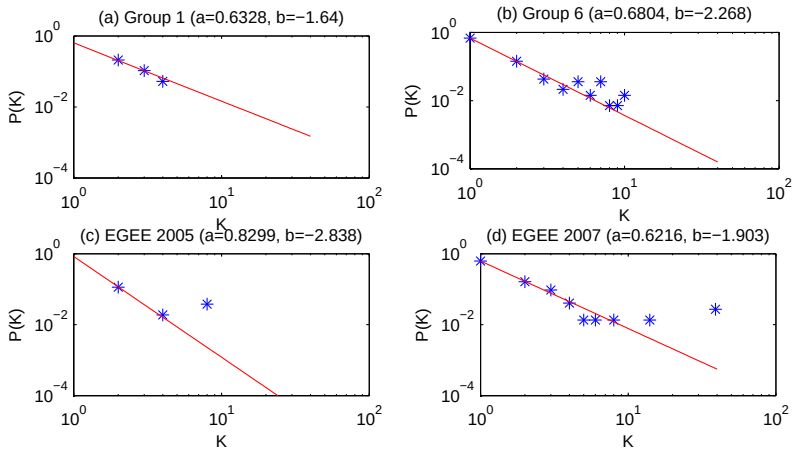

Fig. 5. The power-law distribution of the number of users k following each dominant user identified in Group 1 and Group 6 of Grid 5000, EGEE 2005 and EGEE 2007

includes past job submissions within the interval of C_{job}. Note, this is different to the method described in Algorithm 1 for checking the socially connected jobs. In Algorithm 1, we consider the absolute value of the time difference, which includes both sides of the current time point on the time axis (as shown in Figure 1).

Algorithm 2 allows us to track the socially connected status of the users on the fly. The number of socially connected jobs between users and the number of followers to a given user are updated in real-time as the jobs flow in.

We use *cosine similarity* [13] to measure the difference between the distribution of followers as obtained by Algorithms 1 and 2. We set $C_{job} = 0.5$ *hour* and $C_{user} = 50\%$. After observing $x\%$ of the whole streaming jobs (x-axis), Figure 6 shows the cosine similarity (y-axis) of online results compared to the offline results using all data. The cosine similarity keeps increasing to value 1 by observing additional jobs. When $x\%$ increase to 100% (all jobs have been studied), the online results are same to the offline results (cosine similarity = 1).

A very promising characteristic of the online algorithm is its ability to track group evolution over time. For example, Figure 6 shows how Group 6 and EGEE 2005 have sudden increase in the number of discovered social groups after processing 50% and 80% of all job flows, respectively. In contrast, Group 1 and EGEE 2007 have stable social groups after processing 5% of all jobs, with little change beyond that point.

The changes of social groups discovered online reflect the variation of usage of HPC system committed by the users. In order to verify the online changes of social relationship shown in Figure 6, we investigate the number of distinct users (y-axis) appearing in $x\%$ of all jobs since start (x-axis) in Figure 7 for the 4 data sets. With the increase of $x\%$ toward 100%, we can see the change of the

Algorithm 2 Online Algorithm of Calculating the Socially Influence Matrix

Input: Streaming jobs $S = \{UserID^t,\ JobTime^t\}$,
the $UserID^t$ submitted a job at time $JobTime^t$
Criterion $C_{job} = 0.5\ hour$
Criterion $C_{user} = 50\%$
Output: Socially Influence Matrix M^t, $M_{ij}^t = \{R_{ij}^t, C_i^t\}$
where R_{ij}^t is the number of $U(i)$'s jobs socially connected to $U(j)$ until t, and C_i^t is the number of jobs of user i until t
Initial: the number of distinct users $Un^t = 0$,
set of distinct users $U = \{\}$,
$M_{ij}^t = \{R_{ij}^t = 0, C_i^t = 0\}$

Maintain M^t
for t=1 to ... **do**
  if $UserID^t \notin U$ **then**
    $U = U \cup UserID^t$
    $Un^t = Un^{t-1} + 1$
  end if
  $i \leftarrow UserID^t = U(i)$
  $C_i^t = C_i^{t-1} + 1$
  $X = \{UserID^q\ |q : JobTime^q >= JobTime^t - C_{job}\}$
    the UserID of streaming jobs arrived within a time-window C_{job}
  for j=1 to Un^t **do**
    if $U(j) \in X$ **then**
      $R_{ij}^t = R_{ij}^{t-1} + 1$
      $R_{ji}^t = R_{ji}^{t-1} + |X^{j,t'}|$
        where $|X^{j,t'}|$ is the number of $U(j)$ in time window of $[max\{t - C_{job}, t' + C_{job}\}, t]$ and t' is the last time when $U(i)$ appeared
    end if
  end for

  Socially connected users
  for j=1 to Un^t **do**
    if $R_{ij}^t / C_i^t > C_{user}$ in M_{ij}^t **then**
      user $U(i)$ is socially connected to user $U(j)$, and
      user $U(i)$ is a follower to the dominant user $U(j)$
    end if
  end for
end for

number of users appeared in the system along with time. Comparing Figure 7 and Figure 6, we find that the curves coincide along x for all 4 data sets. For example, Group 6 and EGEE 2005 have more new users after 50% and 80% of all job flows. The number of users in Group 1 and EGEE 2007 grow slowly after 5% of all jobs.

We are still interested in how fast the similarity can reach a stable state. Figure 6 shows that our online algorithm can converge and reach a stable state

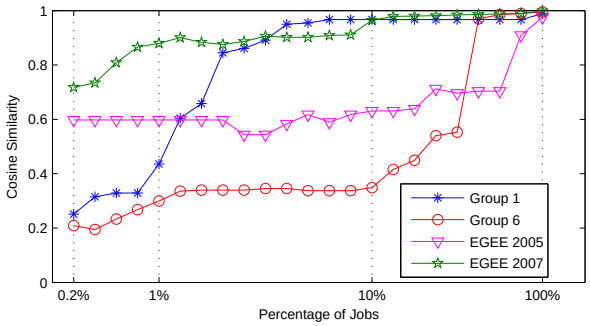

Fig. 6. Online Learning Convergence for Group 1, Group 6, EGEE 2005 and 2007

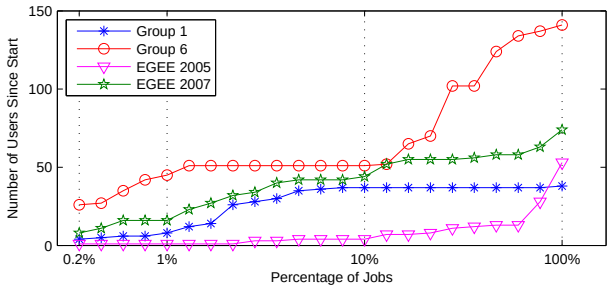

Fig. 7. Online Users Count for Group 1, Group 6, EGEE 2005 and EGEE 2007

quickly by using a small number of jobs. For example in Group 1, we see that the user population (number of distinct users as shown in Figure 7) is stable within 1% of job flows, while the cosine similarity (in Figure 6) reaches the stable state within about 0.4% of all jobs. When the user population of Group 1 changes dramatically from 1% to 7%, Figure 6 shows that the cosine similarity follows the changes quickly and reaches a stable state closing to 1.

6 Related Work

Many generally available workload traces, such as [1–3], have been used to study design alternatives for resource scheduling algorithms, resource capacity planning in both grid cluster and cloud environments, building performance modeling, etc. Most of these studies use—as input—the arrival pattern of individual jobs and their resource requirements (e.g., [8]). Only recently, Iosup *et al.* [5] presented their first investigation of the grouping of jobs by look at their submission patterns and their impact on computing resource consumption. More recently, Ostermann *et al.* [9] studied job flows focusing on how sub-jobs are submitted in

parallel or in sequence [9]. Unlike existing work, we investigate job submission patterns from the perspective of social connections (e.g., group association, virtual organization, etc.). Especially in cloud environments, the ability to predict future demands is critical to managing the underlying computing resources.

Community extraction has been commonly studied as a graph problem. A graph is constructed by taking users/persons as nodes and connecting two nodes if they are correlated in some activities. A community discovered in a graph is a subgraph including a group of nodes and their edges, where the nodes have high similarity with each other. Two approaches have been used to identify the subgraphs. One is based on a clustering method, weighted kernel k-means, that groups together the nodes that are similar to each other by measuring a type of random walk distance [14]. The other approach uses graph partitioning algorithm (or called spectral clustering) to cut the graph into a set of subgraphs by optimizing the cost of cutting edges under the normalized cut criterion [11].

The most relevant work to our paper is the Mixed User Group Model (MUGM) proposed by Song *et al.* [12]. MUGM forms the groups of users through characterizing each user by job clusters, which were obtained by using CLARA (also known as k-medoids) clustering method [6] on the whole jobs. There are two difficulties for applying MUGM on analyzing HPC workloads. First, using clustering method to group jobs requires the computation of job similarities, which is difficult for jobs described by mixture of features. Second, representing users based on job clusters cannot be easily adapted as an online technique, because job clusters need to be computed on the whole data before analyzing the user groups. Our approach identifies social groups by measuring their tasks' submission behavior. Our approach does not need the computation of job similarities and can be efficiently used in an online fashion.

7 Conclusions & Future work

This paper identifies and validates the existence of social influence on HPC workloads. We suspect that this influence stems from how HPC applications are developed and run. Given its potential importance on job scheduling and resource management, we proposed a method to discover *socially-connected users* based on measuring the proportion of *socially-connected jobs* they have. We show the existence of a social graph that is characterized by a pattern of dominant users and followers. We applied this proposed method to traces from Grid 5000 and EGEE. In the Grid 5000 workload, we consistently found that around half of the users have *followers* irrespective of how the thresholds C_{job} and C_{user} are adapted. We develop both an offline and a fast-converging online algorithm to implement our proposed method. The online version was shown to require a small number of jobs (approximately, 1%) to discover the social groups.

Identifying dominant users and their followers may have profound implication on the predication of workload and, consequently, on resource demand patterns. Thus, our future work will focus on quantifying social characteristics of the discovered user connections and using this quantitative information to improve the

prediction of workload arrival patterns and resource demands. The prediction will enable the development of new algorithms in job scheduling, resource utilization, and resource capacity planning.

Acknowledgments

We would like to thank Nicolas Capit, Dr. Franck Cappello, and Dr. Olivier Richard for their help in providing Grid 5000 traces, and thank Prof. Cecile Germain-Renaud for her help in providing EGEE traces.

References

1. Google Cluster Data , http://code.google.com/p/googleclusterdata/
2. Grid Workloads Archive , http://gwa.ewi.tudelft.nl/pmwiki/
3. Parallel Workloads Archive , http://www.cs.huji.ac.il/labs/parallel/workload/
4. Barabasi, A.L., Alberta, R.: Emergence of scaling in random networks. Science 286, 509–512 (1999)
5. Iosup, R., Jan, M., Sonmez, O., Epema, D.: The characteristics and performance of groups of jobs in grids. In: In Euro-Par, volume 4641 of LNCS. pp. 382–393. Springer-Verlag (2007)
6. Kaufman, L., Rousseeuw, P.: Finding Groups in Data: an introduction to cluster analysis. Wiley (1990)
7. Lin, Y.R., Sundaram, H., Chi, Y., Tatemura, J., Tseng, B.L.: Blog community discovery and evolution based on mutual awareness expansion. In: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence. pp. 48–56. WI '07 (2007)
8. Mishra, A.K., Hellerstein, J.L., Cirne, W., Das, C.R.: Towards characterizing cloud backend workloads: insights from google compute clusters. SIGMETRICS Performance Evaluation Review 37(4), 34–41 (2010)
9. Ostermann, S., Prodan, R., Fahringer, T., Iosup, R., Epema, D.: On the characteristics of grid workflows. In: CoreGrid Technical Report TR-0132 (2008)
10. Pacini, F.: Job Description Language HowTo (2001), http://www.infn.it/workload-grid/docs/DataGrid-01-TEN-0102-0_2-Document.pdf
11. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 888–905 (1997)
12. Song, B., Ernemann, C., Yahyapour, R.: User group-based workload analysis and modeling. In: In Proceedings of the International Symposium on Cluster Computing and the Grid (CCGRID). pp. 953–961 (2005)
13. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley Longman Publishing Co., Inc. (2005)
14. Yen, L., Vanvyve, D., Wouters, F., Fouss, F., Verleysen, M., Saerens, M.: Clustering Using a Random Walk Based Distance Measure. In: Proc. of the 13th Symposium on Artificial Neural Networks (ESANN). pp. 317–324 (2005)