HPC Usage Behavior Analysis and Performance Estimation with Machine Learning Techniques

Hao Zhang[1], **Haihang You**[2], **Bilel Hadri**[2], **and Mark Fahey**[2]
[1]Department of Electrical Engineering and Computer Science,
University of Tennessee, Knoxville, TN 37996, USA
[2]National Institute for Computational Sciences,
Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

Abstract—*Most researchers with little high performance computing (HPC) experience have difficulties productively using the supercomputing resources. To address this issue, we investigated usage behaviors of the world's fastest academic Kraken supercomputer, and built a knowledge-based recommendation system to improve user productivity. Six clustering techniques, along with three cluster validation measures, were implemented to investigate the underlying patterns of usage behaviors. Besides manually defining a category for very large job submissions, six behavior categories were identified, which cleanly separated the data intensive jobs and computational intensive jobs. Then, job statistics of each behavior category were used to develop a knowledge-based recommendation system that can provide users with instructions about choosing appropriate software packages, setting job parameter values, and estimating job queuing time and runtime. Experiments were conducted to evaluate the performance of the proposed recommendation system, which included 127 job submissions by users from different research fields. Great feedback indicated the usefulness of the provided information. The average runtime estimation accuracy of 64.2%, with 28.9% job termination rate, was achieved in the experiments, which almost doubled the average accuracy in the Kraken dataset.*

Keywords: Performance Prediction, Usage Pattern Analysis, Recommendation System, User Support, Machine Learning

1. Introduction

High Performance Computing (HPC) is becoming increasingly important by addressing various applications related to science, engineering, service, commerce, security and defense [15]. To improve productivity, more laboratory researchers are starting to use supercomputers to solve large problems in their research fields. However, most researchers without a computer science background or help from HPC consultants have difficulties productively using the supercomputing resources. Programming for a large HPC system requires more experience than for a desktop computer. First, a user needs to know the system-specific information, such as what software packages are supported by the HPC system and how many compute cores are appropriate for a specific job on the system. Furthermore, to submit a job to an HPC system, a user is usually requested to estimate the runtime of a job for system scheduling. In general, the job runtime estimate is very inaccurate. According to the historical usage data of the world's fastest academic Kraken supercomputer, the average accuracy of the estimated runtime is less than 38% for each job category, and around 32.5% for the entire dataset, which is shown in Figure 1. An inaccurate estimate always causes negative effects. An underestimation increases the risk that a job is forcefully terminated by an HPC system before its completion. On the other hand, an overestimation of the job runtime usually results in a longer queuing time. In both cases, the productivity of an HPC user is damaged.

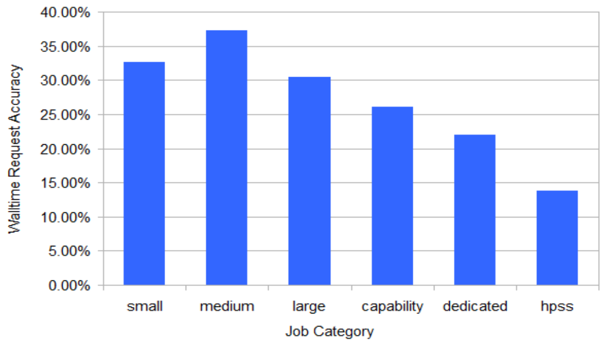


Fig. 1: Average accuracy of the requested walltime for each job category in Table 1.

In this work, we investigated the usage behavior patterns of the Kraken supercomputer, which is the world's fastest petascale academic supercomputer and the 11th on the latest Top500 list. We also developed a knowledge-based recommendation system to optimize user service and resource allocation with the purpose of improving user productivity at the application level. Historical usage logs of the Kraken supercomputer were used to analyze the underlying usage patterns, which were collected from January 2, 2011 to July 21, 2011. We implemented six unsupervised machine learning algorithms to identify usage behaviors, and applied three validation measures to compare the clustering algorithms and determine the appropriate number of behavior categories.

Table 1: Manual classification of jobs in the Kraken supercomputer, which is determined by the number of compute nodes requested by a user.

Categories	Compute Cores			$\text{Walltime}_{\text{max}}$
	Min	Max	Percentage	(hours)
Small	1	512	0-0.45	24.0
Medium	513	58,192	0.45-7.26	24.0
Large	8,193	49,536	7.26-43.88	24.0
Capability	49,537	98,352	43.88-87.12	48.0
Dedicated	98,353	112,896	87.12-100	48.0
HPSS	N/A	N/A	N/A	24.0

The correct jobs were grouped into seven categories, each of which represented a usage behavior of an HPC system. The clustering results can be used by an HPC center to better understand its user community and provide customized services to different groups of users with different types of job submissions. Moreover, considering the fact that HPC systems with different architectures perform differently on the same type of jobs, performance comparisons across different HPC systems provide the potential to identify which architecture is superior to execute which type of jobs. Then, based on the statistics in each behavior category, a knowledge-based recommendation system was developed to provide users with instructions about choosing software packages, setting job parameter values, and predicting job performance. The proposed system enables research communities to share usage experience of a supercomputer, and researchers may improve their productivity by looking at similar jobs successfully executed by other users in the same research area on the same HPC system.

The rest of the paper is organized as follows. Section 2 reviews existing work. Section 3 describes the historical usage data of the Kraken supercomputer, and discusses data preprocessing and feature extraction methods. The unsupervised machine learning algorithms and cluster validation measures are introduced in Section 4 for usage behavior analysis. The proposed recommendation system is also described in this section. Then, Section 5 presents the experimental results on Kraken. Finally, Section 6 concludes the paper.

2. Related Work

Although workload modeling was widely researched [19], only a few studies were reported to address the task of usage behavior analysis. A naive approach to solve this problem is to manually classify a job, based on the number of the requested compute nodes of the job, which is adopted by most HPC centers for job scheduling, such as the Kraken supercomputer [11] as shown in Table 1. Another statistical approach was introduced in [7] to characterize job behaviors and identify the most active users. Wolter [18] experientially classified supercomputer users into three groups. Song [14] proposed a mixed user group model to classify HPC users and analyze workload traces for job scheduling. However, the number of the usage categories was predefined, and the author did not explicitly validate the classification results.

Another related work is to predict the runtime of a job using historical data. The most widely used methods, such as in [10], [16], were directly based on the instances in the historical usage data, which estimated job runtime using the average runtime of a set of jobs with the highest similarity. However, a petascale supercomputer usually has millions of job submissions. It becomes inefficient or even infeasible for instance-based approaches to save all instances. Some model-based algorithms were also proposed for runtime prediction, including methods using artificial neural networks [8] and template-based approaches with greedy and genetic algorithms [13]. However, artificial neural network, as a black box model, failed to explicitly unveil the underlying structures of usage behaviors. In template-based approaches, the templates had to be defined manually, which are almost infeasible for a dataset with millions of samples.

3. Usage Data Processing

In this work, we looked at the historical usage data of the Kraken supercomputer [11], which is managed by the National Institute for Computational Sciences (NICS) at the Oak Ridge National Laboratory. Kraken provides a petascale computing environment fully integrated with the Extreme Science and Engineering Discovery Environment with access to the Cray XT5 system. The Kraken supercomputer consists of 18,816 compute sockets, 147 terabytes of memory, and 3.3 petabytes of storage. Access to computing resources is managed by the Portable Batch System, and job scheduling is handled by the Moab. The Lustre file system is used to support I/O operations.

Usage behavior patterns were analyzed mainly based on the historical data of jobs that were submitted to the Kraken supercomputer. Some instances of usage log entries are listed in Table 2. We also collected account information to determine to which research field each user belongs, and conducted surveys to gather specific information, such as whether a user was an HPC expert. Because most users are not HPC experts or with help from HPC consultants, it is unavoidable that their programs contain runtime errors and exit before completion. On the other hand, due to the lack of experience to run a job on a supercomputer, it is also very common that users underestimate job runtime, and a job is forcefully terminated by the HPC system. If a job neither completes correctly nor provides meaningful results, it is considered as an incorrect job. On the contrary, a job is defined *correct*, if it belongs to the set:

$$\begin{aligned}\mathcal{J}_c = \{j \mid (j.mem\_used \neq 0) \wedge (j.walltime > 30) \\ \wedge\ (j.walltime < j.walltime\_req)\}\end{aligned} \quad (1)$$

where "." represents attribute relationship, and the unit of the attribute *walltime* is second. Intuitively, a job is incorrect if

Table 2: A typical log file that records the usage activities of the Kraken supercomputer. This exemplary log file contains five instances. Each row represents a job submission, and each column denotes an attribute of a job.

job_id	user_name	account	nproc	mem_used	submit_time	start_time
0000001.nid00016	user1	U1-INDEX001	12	7588	2011-01-27 08:10:13	2011-01-27 09:26:03
0000002.nid00016	user2	U2-INDEX001	1	142664	2011-03-28 14:15:50	2011-03-28 14:16:14
0000003.nid00016	user1	U2-INDEX001	1032	16276	2011-04-16 01:28:41	2011-04-16 11:51:30
0000004.nid00016	admin	SUPPORT	288	11836	2011-05-31 09:43:51	2011-06-04 21:00:20
0000005.nid00016	user1	U1-INDEX002	98304	71812	2011-07-05 17:01:08	2011-07-07 11:11:13

end_time	walltime_req	walltime	cpu_hours	queue	type	software	research_area
2011-01-27 09:36:14	00:30:00	00:10:11	3.22	small	batch	—–	Physics
2011-03-28 14:35:01	05:30:00	00:18:47	0.0658	hpss	batch	wrf	Earth Sciences
2011-04-17 11:51:56	24:00:00	24:00:27	24775.74	medium	batch	mpcugles	Physics
2011-06-05 21:00:57	23:59:59	24:00:37	6914.96	small	interactive	—–	Benchmark
2011-07-08 13:11:34	32:00:00	26:00:21	2556477.44	capability	batch	gadget	Earth Sciences

1) it doesn't consume any memory, which indicates that the job fails to start executing, possibly due to some compiling problems, including the situation that the job is compiled on another HPC system with a different architecture; or 2) the job quits right after its execution, which is possibly caused by runtime errors, such as segmentation faults; or 3) the job is terminated by the HPC system, because it consumes the requested walltime. The second literal in (1) also contributes to remove the "Hello World" jobs and simple testing jobs, in which case a user is a learner or does not care about job runtime. Because incorrect jobs are always misleading, it is necessary to remove them to improve the performance of job classification and runtime estimation. On the other hand, it should be noted that the definition of a correct job does not remove all incorrect jobs, such as a job with runtime errors at the end of its execution. The rest of the incorrect jobs are considered as noise, or instances with errors.

A feature is a distinctive characteristic of an object, such as a job, which is often represented as a function of the object's attributes. In this work, four features are selected or extracted from job attribute space, which are defined as:

$$\begin{aligned} &N_n(j) = \log(\, j.nproc \,/\, C_n\,) \\ &M_u(j) = \log(\, j.mem\_used\,) \\ &T_q(j) = \log(\, j.start\_time - j.submit\_time\,) \\ &T_r(j) = \log(\, j.end\_time - j.start\_time\,) \end{aligned} \tag{2}$$

where N_n, M_u, T_q and T_r are the features encoding the information of number of allocated nodes, memory used, job queue time and job runtime, respectively; and C_n represents the number of cores on each node, which is 12 on the Kraken supercomputer. Because job attributes cover a large range of values, logarithmic scale is used to represent each feature to reduce a wide range to a manageable and comparable size.

4. Usage Pattern Modeling

We remove the jobs submitted by system administrators that typically perform system management tasks, such as system updating or reservation removing. Moreover, jobs in the HPSS queue and interactive jobs are also intentionally removed. HPSS jobs often start executing right after being submitted, without any compute nodes involved. Interactive jobs provide a user interactive access to compute resources, which are commonly used for debugging. At last, we manually classify the jobs requiring over 50,000 cores into the group of massive jobs. According to our survey, all users with very large job submissions are HPC experts or work with HPC consultants. Formally, the job dataset used in this work is defined as:

$$\begin{aligned} \mathcal{J} = \{\, j \mid & (j \in \mathcal{J}_c) \wedge (\, j.nproc < 50000\,) \\ & \wedge (\, j.queue \neq \text{hpss}\,) \wedge (\, j.type \neq \text{interactive}\,)\,\} \end{aligned} \tag{3}$$

4.1 Cluster Analysis and Validation

To investigate the underlying structures of usage patterns, cluster analysis is applied on job set $\mathcal{J}$. Cluster analysis is an unsupervised learning technique, which groups objects with similar attributes into respective categories. In cluster analysis, we do not know either which job belongs to which category or the number of categories. Six clustering methods, in three categories, are applied to analyze usage behaviors:

- Partitioning clustering, including k-means [1] and Partitioning Around Medoids (PAM) [17];
- Hierarchical clustering, including DIvisive ANAlysis clustering (DIANA) [5] and Unweighted Pair Group Method with Arithmetic Mean (UPGMA) [9];
- Artificial Neural Network (ANN) based clustering, including Self-Organizing Feature Map (SOFM) [6] and Self-Organizing Tree Algorithm (SOTA) [4].

Cluster validation evaluates the performance of a clustering result by comparing it with other ones that are generated by other clustering methods, or by the same method but with different parameters, such as different number of clusters. In this work, three internal validation measures, along with domain knowledge, are used to choose the best clustering methods and determine the number of clusters that is most appropriate for the dataset:

- Connectivity [2] measures cluster connectedness, with a value in the range $[0, \infty]$. A lower value indicates a better performance.
- Dunn index [3] is the ratio of the smallest distance between objects in different clusters to the largest intra-cluster distance. The value of Dunn index lies in $[0, \infty]$, and a greater value indicates a better performance.
- Silhouette width [12] measures the degree of confidence in the clustering assignment of an object. Its value lies in $[-1, 1]$ with a greater value indicating a better clustering performance.

4.2 Information Recommendation

It is clear that jobs have a wide range of system demands and might perform differently on the same HPC system. For example, computation-intensive and data-intensive jobs might have significantly different runtime, even if they request the same number of compute nodes. The proposed knowledge-based recommendation system is based on the plausible assumption that job submissions in the same research field tend to perform more similarly, since jobs submitted by researchers in the same field are more possible to address similar problems with similar algorithms using similar software and packages. Moreover, jobs in the same clustered categories also tend to behave more similarly due to the similarity of their attributes. In consideration of both assumptions, we predict the parameters of a target job based on the information of the jobs that are in the same category and same research field as the target job.

In the first step, customized information is provided to a user by the recommendation system, including the distribution of the jobs, the statistical information of several job features, and a list of the most frequently used software and packages in the given research field. Jobs in a research field conforms to the multinomial distribution given job categories. A bar graph is used to intuitively represent this distribution. Then, the statistical summaries of several features are graphical displayed using box plots. Each box plot represents the distribution of a feature, its central value, and variability. The displayed features include N_n and M_u, i.e., the number of requested compute nodes and consumed memory. At last, a list of software and packages is provided, which is obtained by ranking the most frequently used software and packages used by previous users in the same research field. In this step, a user is required to determine the number of requested cores and memory, under the guidance of the recommendation system.

In the second step, after the number of requested cores and memory are decided, the features N_n and M_u are computed according to (2). Then, the queuing time and runtime of the target job are predicted based on the job distributions and the job statistics in each behavior category. More precisely, our goal is to predict the feature vector $\boldsymbol{z}_j = \{T_q, T_r\}$. Let $\boldsymbol{y}_j = \{M_u, Nn\}$ be the feature vector that is determined in the first step by a user in field r_j. Then, the distance between $\boldsymbol{y}_j$ and the center of the behavior category k can be computed using Euclidean distance: $d_{jk} = \|\boldsymbol{y}_j - \mathrm{E}[\boldsymbol{y}_{jk}]\|$, where $\mathrm{E}[\boldsymbol{y}_{jk}]$ is the feature vector expectation of the jobs in the behavior category k and the research field r_j. Then, the target feature vector $\boldsymbol{z}_j$ can be computed by:

$$\boldsymbol{z}_j = \frac{1}{Z_j(\boldsymbol{y}_j, \boldsymbol{\theta}_j)} \sum_{k=1}^{K} \frac{\theta_{jk}}{d_{jk} + \sigma} \mathrm{E}[\boldsymbol{z}_{jk}] \tag{4}$$

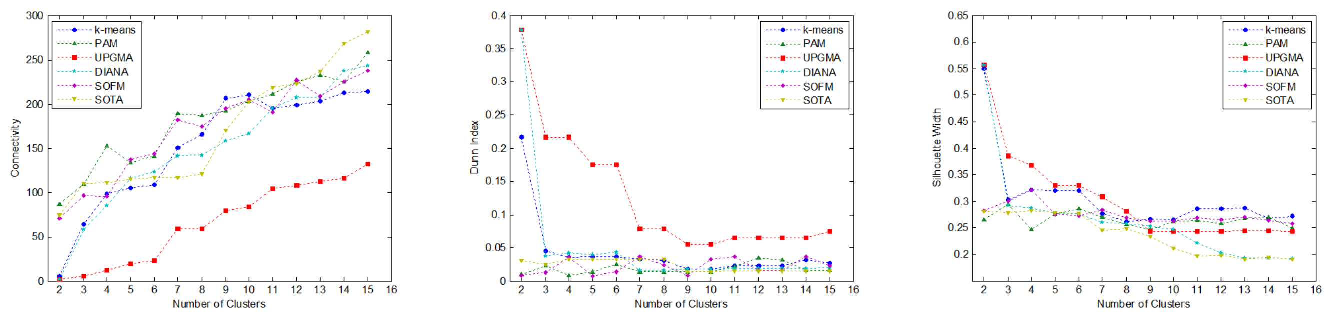
where θ_{jk} is the probability that job j belongs to category k; σ is a small positive number to avoid a zero denominator; and $Z_j(\boldsymbol{y}_j, \boldsymbol{\theta}_j)$ is a normalizing factor. The feature vector $\boldsymbol{z}_j$ is computed using a mixture of corresponding cluster means. The mixture coefficient considers both the clustering result of the target job, which is represented by the distance to each cluster center, and the prior knowledge $\boldsymbol{\theta}_j$ that is the distribution of the jobs in the target job's research field. Both elements are important for job classification. While the distance of a job to a cluster center determines the probability that the job belongs to the cluster, the job distribution in a research field, as a prior, prefers the cluster containing a larger number of jobs that are previously observed. After the feature vector $\boldsymbol{z}_j$ is calculated, the queuing time can be computed by: $j.qu\hat{euing} = 10^{T_q}$. In most cases, users care more about job runtime, which is a required parameter by most supercomputers to submit a job. According to our experiments, the job runtime can be better estimated by:

$$j.run\hat{t}ime = 10^{(1+\alpha)T_r + \beta} \tag{5}$$

where $\alpha, \beta \in [0, 1]$. A good job runtime estimation can be computed with $\{\alpha, \beta\} = \{0.05, 0\}$, and a safer estimation can be obtained with $\{\alpha, \beta\} = \{0.05, 0.3\}$, which can be applied to request system walltime. The coefficient $(1 + \alpha)$ of T_r considers the fact that, for a larger job with more compute resources and longer runtime, users often tries to make the job safer (i.e., with a higher probability to successfully complete the job), by intentionally requesting a longer system walltime. If a large job is forcefully terminated by the system before its completion, it often costs much more than a small job.

5. Experimental Results

We used the historical usage data that were collected from January 2 to July 21, 2011, after Kraken was upgraded to the Cray XT5 system. During the time of data collection, 321,290 jobs in 24 different research fields were submitted to



(a) Connectivity measure (b) Dunn index measure (c) Silhouette width measure

Fig. 2: Internal cluster validation of the clustering methods with different number of clusters.

Kraken by 843 researchers from 367 accounts. The Kraken dataset was first preprocessed to get job set $\mathcal{J}$. In this step, 112265 jobs were removed from the original Kraken dataset, including 102206 incorrect jobs (termination-rate = 31.8%). Then, feature vectors were computed for the rest of the jobs according to (2), and related information, including research field and software name, was documented.

5.1 Cluster Analysis and Validation

Before clustering the Kraken dataset, a cluster was manually defined as the massive job category that contains jobs requiring more than 50,000 compute cores, which contains 168 job submissions in the entire preprocessed dataset.

Six clustering approaches were applied to cluster usage behaviors, and three internal validation measures were employed to compare the clustering performance and to find the appropriate number of clusters. The performance evaluation results are depicted in Figure 2a, 2b and 2c, respectively. A most important conclusion is that the UPGMA algorithm performs consistently better than other clustering methods when using less than nine clusters, which is indicated by all internal validation measures. We can also observe that other clustering techniques, if the number of clusters is greater than three, have similar performance but are generally worse than the UPGMA algorithm. Thus, the UPGMA algorithm, an agglomerative hierarchical clustering algorithm, was selected to cluster usage behaviors.

The validation measures were also applied to determine the appropriate number of clusters. For all clustering algorithms, the connectivity measure in Figure 2a indicates that, with the increase of the number of clusters, the clustering performance tends to decrease. Similarly, both the Dunn index in Figure 2b and the silhouette width in Figure 2c indicate that clustering performance cannot be improved by simply increasing the number of clusters. We can also observe that, for the k-means, PAM and UPGMA algorithms, two is the best choice for the number of clusters, which is supported by the connectivity measure and the Dunn index measure, and partially by the silhouette width measure. On

Table 3: Algorithmic mean of each job feature in different categories in the Kraken dataset.

Feature	c1	c2	c3	c4	c5	c6	c7
E(T_q)	4.24	2.62	2.87	1.89	4.65	3.74	4.27
E(T_r)	4.19	3.63	3.22	2.38	4.15	2.64	3.22
E(M_u)	4.05	4.03	5.11	4.09	4.17	4.12	5.07
E(N_n)	0.19	0.26	1.01	0.98	1.51	1.51	3.90

the other hand, in order to increase the resolution of the job clustering results, we prefer a larger number of clusters. For the chosen UPGMA algorithm, the clustering performance almost consistently declines with the increase of the number of clusters. But the performance of the algorithm does not change greatly with the number of clusters lying between three and six. To balance clustering performance and clustering resolution, six clusters were selected, in which case, we increased the clustering resolution by sacrificing some clustering performance. The resulted dendrogram with six clusters is depicted in Figure 4.

The algorithmic means of job features in each category are listed in Table 3. Several interesting phenomena should be noted. First, the behavior category c3 can be considered as the set of data-intensive jobs. Jobs in this category use significant memory but relatively less number of compute nodes. Second, the jobs in the manually defined category c7 are greatly different from the jobs in other categories, which are both computational and memory-intensive. However, jobs in this category do not have the longest queuing time, because the massive jobs have a high priority on the Kraken supercomputer, and researchers running such jobs usually reserve the compute resources in advance. These massive jobs do not have the longest job runtime either. Besides the fact that the massive jobs use considerable amount of resources that can speed up job execution, this phenomenon can be partially explained by the fact that most users running such jobs have rich HPC knowledge. In most cases, the massive jobs are optimized. Third, if we only consider the

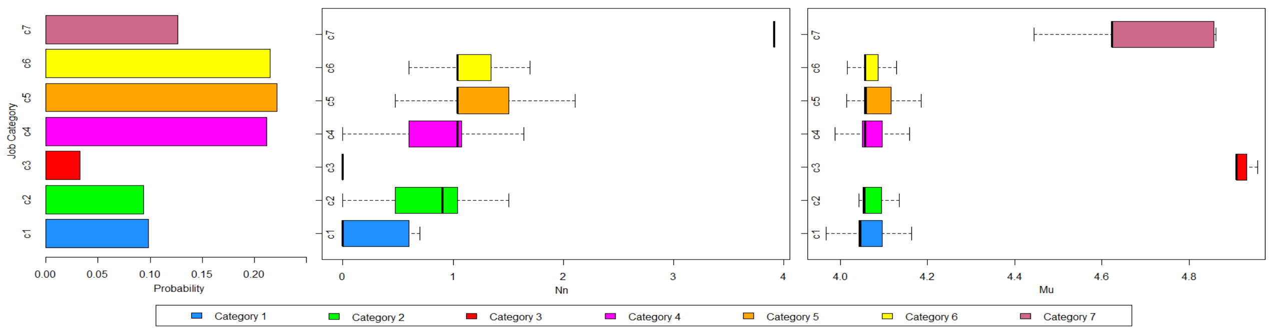


Fig. 3: Customized information of the jobs in the research field of Astronomical Sciences, which is generated in the first-round recommendation. The bar graph (left) represents the distribution of the previously submitted jobs on Kraken. The box plot (center) summaries the statistics of the number of compute nodes requested by the jobs in each category in log scale. The box plot (right) represents the statistical summary of the consumed memory of the jobs in each category in log scale.

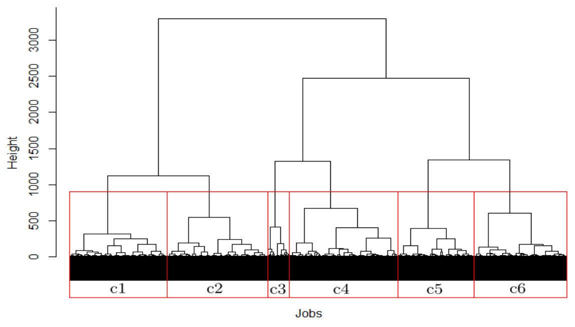


Fig. 4: Dendrogram generated by the UPGMA algorithm with six categories on the Kraken dataset.

requested compute nodes and the consumed memory to classify jobs, we can combine the category pair: c1 and c2, c3 and c4, c5 and c6 in to larger clusters, due to the similarity of the features in each category pair. This phenomenon is well supported by the dendrogram. If the UPGMA algorithm is set to stop with three clusters, the dendrogram will be cut at the height of 2000, as shown in Figure 4. Fourth, jobs in the categories c1 and c2 have the longest runtime. Besides the fact that these jobs use less compute resources, another possible explanation is that users executing such jobs usually have less HPC experience and their jobs are not well-tuned.

5.2 Information Recommendation

To demonstrate how the proposed recommendation system instructs a user to select appropriate job parameters and predict job performance, a real case is used as an example, in which case, a researcher planned to solve a problem in the field of Astronomical Sciences. In the first-round recommendation, the system first provided a list of packages that were most frequently used on Kraken to solve problems in Astronomical Sciences, such as GADGET, YT, SSES, and CHIMERA. After comparing the packages, the researcher decided to use Gadget, which is a cosmological smoothed particle hydrodynamics (SPH) simulator. The proposed system also provided the researcher with customized information about the job statistics in Astronomical Sciences, as shown in Figure 3. Some important phenomena were observed by the researcher. First, more large jobs in c4, c5, c6 and c7 were executed in this field. Second, almost all massive jobs in c7 used considerable resources with similar number of compute nodes but widespread memory. Third, only a few jobs were data-intensive which generally consumed a very small number of compute nodes but significant amount of memory. Fourth, the non-massive non-data-intensive jobs, in c1, c2, c4, c5 and c6, consumed similar amount of memory, as shown by the right box plot. The medians of the number of requested compute nodes of jobs in c4, c5 and c6 were very close, as indicated by the center box plot in Figure 3. Based on these observations, the researcher decided to request 12 GB memory ($\hat{M}_u \approx 4.1$) and 12 compute nodes ($\hat{N}_n \approx 1.1$) for the job.

In the second-round recommendation, the values of T_q and T_r were computed according to (4), given $\hat{M}_u$ and $\hat{N}_n$. The architecture of the recommendation system and the estimating process were transparent to users. All feedback the researcher received from the system was the estimated queuing time, runtime, and a safe runtime, which can be used to request system walltime on the Kraken supercomputer. In this case, the estimated queuing time was 7079 seconds ($z_{Tq} = 3.85$); the estimated runtime was 5624 seconds ($z_{Tr} = 3.57$); and the estimated safe runtime was 11221 seconds. After successfully compiling the program on Kraken, the researcher used the estimated runtime as the requested system walltime to execute the job. After waiting in the queue of type *small* for 11175 seconds on Kraken, the job started executing, and the actual runtime was 5049 seconds, which was smaller than but quite close to the job estimated runtime provided by the proposed recommendation system.

In order to quantitatively measure the accuracy of the runtime estimation, we applied the Walltime Request Accuracy (WRA) for a correct job j in $\mathcal{J}_c$, which is defined as:

$$WRA(j) = \frac{j.walltime}{j.walltime\_req} \times 100\% \tag{6}$$

In the evaluation, 127 jobs submitted by seventeen researchers from different research fields were used to test the proposed system. The researchers were not HPC experts or worked with HPC consultants, and the jobs were moderately optimized. According to their feedback, all researchers considered that the information provided by the recommendation system was helpful, especially the job statistics in a research field. For job runtime estimation, when the researchers were asked to use the estimated runtime provided by the recommendation system as the requested walltime, 28.9% jobs were terminated in force by the Kraken supercomputer, due to running out of requested walltime. For the correct jobs, an average WRA of 64.2% was achieved. Comparing to the average WRA of 32.5% with the termination rate of 31.8% in the Kraken dataset, the estimation performance was greatly improved. If the users were asked to use the safe runtime estimate to request system walltime, only 12.2% jobs exceeded their requested walltime. But the average WRA decreased to 33.3%. In this case, although the resulted average WRA was similar to the average WRA in the Kraken dataset, the termination rate was greatly decreased. In order to quantitatively measure the error of the estimated queuing time, the mean absolute percentage error (MAPE) was applied, which is defined as:

$$MAPE = \frac{1}{T}\sum_{t=1}^{T}\left|\frac{j.qu\hat{e}uing - j.queuing}{j.queuing}\right| \tag{7}$$

where T is the number of testing instances, and $j.qu\hat{e}uing$ and $j.queuing$ are the estimated and actual queuing time, respectively. The resulted MAPEs were 67.2% and 105.8% using the estimated runtime and the safe runtime to request system walltime, respectively. This result indicated that the queuing time is harder to estimate, which is heavily dependent on the job queue status. Fortunately, the researchers, like most HPC users, did not much care about the queuing performance according to their feedback.

6. Conclusion

Usage behaviors of the Kraken supercomputer were systematically investigated and a knowledge-based recommendation system was developed to improve job performance and user productivity at the application level. Besides manually defining a category for massive job submissions, six behavior categories were identified with the UPGMA algorithm, which presented the most promising performance on the Kraken dataset. Then, a knowledge-based recommendation system was developed based on the identified behavior categories and job statistics. The proposed recommendation system is able to: 1) provide customized information to help a user determine the software packages, and the number of compute nodes and amount of memory to request for a job, 2) predict job queuing time and runtime, and estimate a safe job runtime to request system walltime. Great feedback from users demonstrated the usefulness of the recommendation system. The average runtime estimation accuracy of 64.2%, along with the job termination-rate of 28.9%, was achieved, which almost doubled the previous average accuracy in the Kraken dataset.

References

[1] K. Alsabti, S. Ranka, and V. Singh, "An efficient space-partitioning based algorithm for the k-means clustering," in *The Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, 1999, pp. 355–359.

[2] L. Deborah, R. Baskaran, and A. Kannan, "Survey on internal validity measure for cluster validation," *International Journal of Computer Science and Engineering Survey*, vol. 1, no. 2, pp. 85–102, Nov. 2010.

[3] J. C. Dunn, "A fuzzy relative of the ISODATA process and its use in detecting compact Well-Separated clusters," *Journal of Cybernetics*, vol. 3, no. 3, pp. 32–57, 1973.

[4] J. Herrero, A. Valencia, and J. Dopazo, "Phylogenetic reconstruction using a growing neural network that adopts the topology of a phylogenetic tree," pp. 226–233, 1997.

[5] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley-Interscience, Mar. 1990.

[6] T. Kohonen, *Self-organized formation of topologically correct feature maps*, 1988, pp. 509–521.

[7] H. Li, D. Groep, and L. Walters, "Workload characteristics of a multi-cluster supercomputer," *Job Scheduling Strategies for Parallel Processing*, vol. 3277, 2004.

[8] J. Li, X. Ma, K. Singh, M. Schulz, B. de Supinski, and S. McKee, "Machine learning based online performance prediction for runtime parallelization and task scheduling," in *IEEE International Symposium on Performance Analysis of Systems and Software*, 2009.

[9] C. D. Michener and R. R. Sokal, "A quantitative approach to a problem in classification," *Evolution*, vol. 11, 1957.

[10] T. Minh and L. Wolters, "Using historical data to predict application runtimes on backfilling parallel systems," in *Euromicro International Conf. on Parallel, Distributed and Network-Based Processing*, 2010.

[11] NICS, "Running jobs on Kraken," http://www.nics.tennessee.edu/node/16, accessed: 11/11/2011.

[12] P. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, Nov. 1987.

[13] W. Smith, I. T. Foster, and V. E. Taylor, "Predicting application run times using historical information," in *Workshop on Job Scheduling Strategies for Parallel Processing*, 1998, pp. 122–142.

[14] B. Song, C. Ernemann, and R. Yahyapour, "User group-based workload analysis and modelling," in *IEEE International Symposium on Cluster Computing and the Grid*, vol. 2, may 2005, pp. 953–961.

[15] Top500, "Application area share for 06/2011," http://www.top500.org, accessed: 11/11/2011.

[16] D. Tsafrir, Y. Etsion, and D. Feitelson, "Backfilling using system-generated predictions rather than user runtime estimates," *IEEE Trans. Parallel and Distributed Systems*, vol. 18, no. 6, pp. 789–803, 2007.

[17] M. van der Laan, K. Pollard, and J. Bryan, "A new partitioning around medoids algorithm," *Journal of Statistical Computation and Simulation*, vol. 73, no. 8, pp. 575–584, 2003.

[18] N. Wolter, M. McCracken, A. Snavely, L. Hochstein, T. Nakamura, and V. Basili, "What's working in HPC: Investigating HPC user behavior and productivity," *CTWatch Quarterly*, vol. 2, no. 4A, 2006.

[19] H. Zhang and H. You, "Comprehensive workload analysis and modeling of a petascale supercomputer," in *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, 2012.