







Automated HPC Workload Generation Combining Statistical Modeling and Autoregressive Analysis

Zechun Zhou , Jingwei Sun  , and Guangzhong Sun 

School of Computer Science and Technology, University of Science
and Technology of China, Hefei, China
zhouzechun@mail.ustc.edu.cn, {sunjw,gzsun}@ustc.edu.cn

Abstract. Understanding the characteristics of workloads is essential to improving the management of a High Performance Computing (HPC) cluster. However, due to the restrictions of privacy and confidentiality, real HPC workloads are rarely open for studying. Generating synthetic workloads that mimic real workloads can facilitate related research, such as cluster planning and scheduling. Thus automated HPC workload generation has long been an active research topic. In this paper, we introduce a workload modeling approach that combines statistical modeling and autoregressive analysis. The model we built can generate complex, realistic HPC workloads with features that clearly describe the scheduling process, including job arrival time and other job attributes that affect scheduling such as job run time and job requested resources. Job arrivals in HPC clusters are generally represented by stochastic processes. In our proposed approach, job arrivals will be generated by a statistical model that consists of multiple Poisson processes with constraints provided by Gamma distribution. Then, we perform autoregressive analysis on the changing trends of job attributes to extract sequence information from historical workload trends that reflect user habits and scheduling habits in the cluster. Our approach generates job attributes based on the extracted sequence information for each job in the generated job arrival sequence. We evaluate the performance of the proposed approach using multiple metrics as well as a real-world use case. Experiments on real workloads from four supercomputing centers validate the effectiveness of the proposed method.

Keywords: Workload generation · Workload characterization · Cluster scheduling · Statistical modeling · Autoregressive analysis

1 Introduction

Understanding the characteristics of workloads is essential to promoting the management of an HPC cluster. These known workload characteristics enable better services that have data dependence on workload status, such as cluster

planning and scheduling [4,34]. However, real HPC workloads are rarely open for studying because of the restrictions of privacy and confidentiality. Due to the lack of publicly available real HPC workloads, synthetic workloads are widely used in HPC research. For example, with synthetic workloads, portable I/O analysis of commercially sensitive HPC applications can be conducted [11]. The evaluation of scheduling algorithms also relies on the large number of synthetic workloads available [24,30]. In particular, learning-based schedulers can be more fully trained on a large number of available synthetic workloads [12,13]. Therefore, automated HPC workload generation has long been an active research topic among HPC providers and researchers [4,23].

To accurately synthesize HPC workloads, characterizing workload patterns is essential. Some of these characteristics are intuitive and straightforward, such as the natural fluctuations in workloads throughout the week. Weekday workloads tend to be heavier than weekend workloads. Besides, there are implicit characteristics, such as workload fluctuations due to user habits [9]. Therefore, it is crucial to construct a detailed model and incorporate a broad set of factors to facilitate the extraction of all workload characteristics. With a comprehensive understanding of these workload characteristics, synthetic workloads can then be generated reasonably.

The classic approach to workload generation is statistical modeling [8,25,35]. Statistical modeling is a type of modeling method based on the regression of a large number of independent data to obtain the statistical distribution of the objective. For example, Poisson regression is a conventional method to simulate job arrivals [4]. It assumes that a large number of randomly arriving jobs usually follow a Poisson distribution. Then based on the time-dependent features, the regression model will fit a Poisson distribution for the job arrivals in each time interval. Other job attributes such as job run time are also typically generated by statistical modeling. The distribution of job run time is usually obtained by the empirical counting of the run time of all jobs in a period [8].

However, relying solely on statistical modeling frequently falls short of accurate workload generation. The primary focus of statistical modeling is on the distribution of workloads, neglecting potential correlations between various workload sequences. In this paper, we combine statistical modeling and autoregressive analysis for automatic HPC workload generation. At first, the job arrivals are obtained by modeling and simulation. Then other job attributes are further generated for the generated job arrival sequence.

In our workload generation approach, a statistical model combining Poisson regression and Poisson-Gamma regression is used to generate job arrivals. The commonly used Poisson regression works well in clusters with large amounts of job arrivals [9]. But in clusters with sparse workloads, the Poisson distribution assumption of job arrival distribution is difficult to be satisfied. Overdispersion of job arrivals occurs in these cases [32], which we deal with by introducing a Poisson-Gamma distribution. The hybrid model of Poisson regression and Poisson-Gamma regression can handle more job arrival distributions than Poisson regression alone.

Other job attributes, such as job requested resource and job run time, are generated by the autoregressive model in our approach. These job attributes are organized as time series in the historical job arrival sequences. In our approach, we perform attentive sequential generation for these job attributes based on autoregressive analysis of serial autocorrelation. Compared with statistical modeling, our model can better reproduce the data correlation in the series [4]. Based on the seasonal scheduling behavior derived from historical workloads, coupled with simulated job sequences provided by the job arrival model, job attributes of each job can be reasonably generated in our approach.

We evaluate the proposed workload generation approach on real HPC workload traces from four different real-world supercomputing centers. Our results show that our model can reveal the characteristics of real workload accurately. The synthetic workload generated by the proposed model reproduces scheduling information in the real workloads. In the evaluation, we use multiple metrics to measure the effectiveness of existing methods as baselines and show the superiority of our model.

The main contributions of this study are summarized as follows:

- We simulate job arrivals based on statistical modeling combining Poisson regression and Poisson-Gamma regression, with which more diverse arrival situations can be accurately represented.
- We conduct an autoregressive analysis to better model serialized workload information so that we can perform attentive sequential generation for job attributes realistically. So that the job attributes can be realistically generated for the simulated job sequence.
- We evaluate the proposed method on real workload traces from four supercomputing centers. All evaluations demonstrate the superiority of using our model to generate synthetic workloads.

Table 1. Specifications of workload traces from four platforms.

	#nodes	#cores/node	#jobs	Time span
Platform A	450	24/28	288K	364 days
Platform B	100	64	248K	364 days
Platform C	416	12	264K	364 days
Platform D	114	28	23K	364 days

2 Preliminary

We conduct our experiments on real traces from four different supercomputing centers. They are Supercomputing Center of University of Science and Technology of China [3], Center for High Performance Computing in Shanghai Jiao Tong

University [1], Shanghai Supercomputing Center [2], and Gansu Supercomputing Center respectively, and are subsequently referred to as platforms A, B, C, and D. The workload trace data from platform A is a typical dataset to conduct our experiments, and the inspiration for building our workload generation model is mainly based on our exploration of it. The workload trace data from the other three platforms are used as supplementary datasets in our evaluation experiments to verify the generalization of our approach. The specifications of traces from these four platforms are shown in Table 1. Among them, the traces of the first three platforms have similar specifications, while the data size of platform D is smaller.

These workload traces are recorded in the Standard Workload Format (SWF) proposed by David Talby and refined through discussions by Dror Feitelson, James Patton Jones, and others [6]. For each job, we mainly pay attention to its user ID, submit time, run time, and requested resources. A complete job sequence plus these job attributes can describe a complete scheduling process. For evaluation of workload generation, the last 10 weeks of each trace will be used as a test window, while all the data will be used to train the workload generation model.

The above platforms used for evaluation experiments mainly carry scientific computing workloads for researchers in various fields [1–3]. They are dominated by these tightly coupled MPI jobs. Common scientific computing software, such as VASP (Vienna Ab initio Simulation Package) [18] and Gaussian [10], often submits a large-scale parallel computing job to the cluster. For reasons such as adaptation or price, users’ main choice for computing resources is the CPU rather than other acceleration devices such as GPU. The computing power of the computing nodes of the above platforms is indeed mainly provided by the CPU. So in this paper, job requested resources refer to the number of requested computing nodes, most of which are computing nodes with multiple CPU cores. Moreover, as the computing model expands, the run time of scientific computing tasks will increase beyond linear corresponding increases [19]. Therefore, the run times of jobs in these workloads vary greatly, with the longest scientific computing jobs running for weeks and the shortest test jobs running for just seconds.

3 Modeling Methodology

The workload generation model proposed in this paper consists of two major components, the job arrival model and the job attribute model. The overview of the modeling framework is shown in Fig. 1.

First, the arrival of jobs within each hour will be described by stochastic processes. The job arrival model employs two regression tasks to model job arrivals. Then, for each generated job with unknown attributes, the job attribute model autoregressively generates job attributes that conform to user habits at the user granularity, including requested resources, run time, job queue, etc.

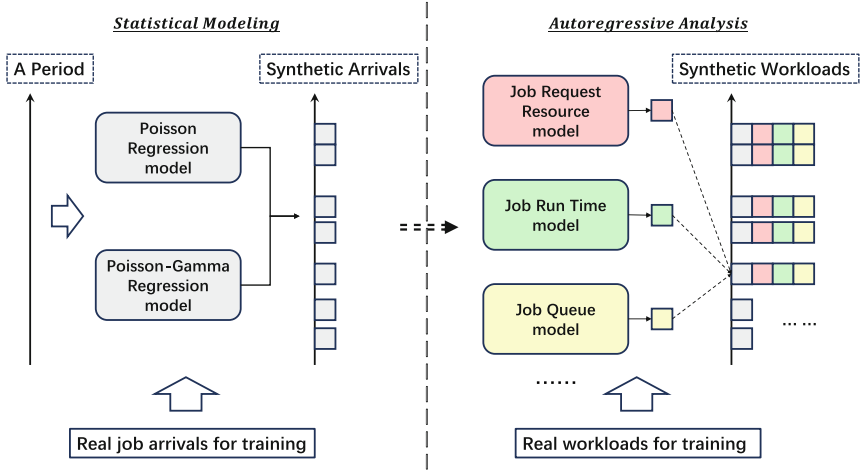


Fig. 1. Overview of modeling framework.

3.1 Job Arrival Model

The job arrival model employs two regression tasks, Poisson regression and Poisson-Gamma regression to model the stochastic process of job arrivals. We treat job arrivals at one-hour intervals within the trace range as random arrivals. The feature vector of each interval is determined by the following three features:

- *HoD*: One-hot code, to indicate this interval belongs to which hour of the day. Its dimension is 24.
- *DoW*: One-hot code, to indicate this interval belongs to which day of the week. Its dimension is 7.
- *WoY*: One-hot code, to indicate this interval belongs to which week of the year. Its dimension is 52.

HoD and *DoW* determine the feature vector of this interval in the regression task. They describe the temporal characteristics of job arrivals on day and week scales, respectively. *WoY* does not participate in the regression task. We use it to calculate a trend correction term for job arrivals that changes with the No. of weeks to make the regression model more generalizable. *WoY* is also used as the basis for us to assign user IDs to jobs within a week.

Poisson Model. Poisson regression is widely used in the simulation of job arrivals [4, 13, 16]. The probability distribution of a single Poisson distribution is: $f(y = k; \lambda_i) = \frac{\lambda_i^k}{k!} e^{-\lambda_i}$, $k = 0, 1, \dots$, and Poisson process assumes that in interval i , it contains y_i events arrive randomly. For interval i , we use feature vector \mathbf{x}_i to describe its temporal features. *HoD* and *DoW* determine a 168×168 one-hot code feature matrix \mathbf{X} . For any feature vector, it can be matched to a row in the feature matrix. Then, the distribution parameters can be expressed

as $\lambda = e^{\mathbf{X}\beta}$, where \mathbf{X} is the feature matrix composed of independent feature vectors, and β is the parameter matrix that needs to be regressed. The log-likelihood function of Poisson regression can be calculated by:

$$l(\beta) = \sum_{i=1}^n (y_i \mathbf{x}_i \beta - e^{\mathbf{x}_i \beta}), \quad (1)$$

where n represents that there are n observations in the training set, and y_i is the number of event arrivals when the corresponding feature vector is \mathbf{x}_i . The partial derivative of the negative log-likelihood function in the direction of β is used as *Loss* for training, and the β parameter can be optimized by minimizing the negative log-likelihood estimation.

Poisson-Gamma Model. In real data, due to problems such as overdispersion and zero-inflation [32], the naive Poisson process is difficult to fully describe the arrivals in the real world. For such data, Its probability distribution can be better represented by a negative binomial distribution, or a Poisson-Gamma mixture distribution: $f(y = k; r, p) = \frac{\Gamma(k+r)}{\Gamma(k+1)\Gamma(r)} p^r (1-p)^k, k = 0, 1, \dots$. For the overdispersed part, negative binomial regression can be chosen instead of naive Poisson regression, by replacing the parameters as follows: $\alpha_i = \frac{1}{r}, \mu_i = \frac{1-p}{\alpha p}$. Then the log-likelihood function of negative binomial regression can be calculated by:

$$l(\alpha, \mu) = \sum_{i=1}^n \left\{ y_i \ln \frac{\alpha \mathbf{x}_i \mu \mathbf{x}_i}{1 + \alpha \mathbf{x}_i \mu \mathbf{x}_i} + \ln \Gamma(y_i + \frac{1}{\alpha \mathbf{x}_i}) - \frac{1}{\alpha \mathbf{x}_i} \ln(1 + \alpha \mathbf{x}_i \mu \mathbf{x}_i) - \ln \Gamma(y_i + 1) - \ln \Gamma(\frac{1}{\alpha \mathbf{x}_i}) \right\}. \quad (2)$$

Hybrid Model. The above two regression models will be trained to convergence taking into account zero inflation of the data. At this point, we have two different models describing job arrivals during each one-hour interval. The Poisson model is more suitable for describing data with sufficient statistics and less interference. The Poisson-Gamma model can describe overdispersed data more accurately. We fuse these two models into a unified hybrid model based on Bayesian generalized additive models [21].

This hybrid model serves as our final job arrival model to simulate job arrivals within each one-hour interval. These job arrivals will be modified based on the correction of the weekly job arrival trend according to the interval's *WoY*. Then a sequence of jobs with just arrival time is generated, $\mathbf{J} = \{j_1, j_2, j_3, \dots\}$. Also based on *WoY*, we count the proportion of job arrivals for each user every week and randomly assign user IDs to all jobs within a week according to the generalized Bernoulli processes, which is based on the multinomial distribution of job arrivals over the user set. Let the user set be $\mathbf{U} = \{u_1, u_2, u_3, \dots\}$, and the entire job sequence \mathbf{J} can be divided into multiple subsequences based on user IDs: $\mathbf{J}^{u_1} = \{j_1^{u_1}, j_2^{u_1}, j_3^{u_1}, \dots\}, \mathbf{J}^{u_2} = \{j_1^{u_2}, j_2^{u_2}, j_3^{u_2}, \dots\}, \dots$.

3.2 Job Attribute Model

The function of the job attribute model is to generate reasonable job attributes for each job in the simulated job arrival sequence $\{j_1, j_2, j_3, \dots\}$. Then we describe the attribute embedding, the autoregressive analysis process, and the attentive sequential workload generation in our job attribute model.

Attribute Embedding. The job attributes recorded in HPC cluster trace include many items, such as requested resources, run time, job queue, job status, etc. The job attributes that are the targets of our modeling generation can be specific to the following two items: (1) Job Requested Resource, indicates the number of computing resources requested by this job, for example, 5 Computing node cores; (2) Job Run Time, indicates the time that this job occupies computing resources while running; Based on these two items, the job scheduling process within an HPC cluster can be clearly constructed. Other miscellaneous items that have a minor impact on the scheduling process are not our modeling generation targets due to the inconsistency in trace record attribute types across platforms. But all items will contribute features to our attribute embedding, let the number of items be k . In order to facilitate attribute generation, we quantize the job attributes, in which the values of these items will be mapped to k discrete sets by clustering. In this way, the job attributes can be uniquely represented by one-hot codes $\in \mathbb{R}^{k \times D}$, where D is the dimension of discrete sets. We encode these items together into an embedding vector to represent the attributes of the job:

$$A(j) = \text{Embed}[R(j), T(j), O(j)], \quad (3)$$

where $R(j)$, $T(j)$, and $O(j)$ respectively represent the requested resources, run time, and other miscellaneous items of job j . $A(j) \in \mathbb{R}^d$ is an embedding vector representing the attributes of job j . From this, job attributes can be generated based on a multinomial distribution over k discrete sets. Our job attribute model is built based on autoregressive analysis.

Autoregressive Analysis. For a user's sequence of jobs sorted by arrival time, job attributes are autocorrelated over time scales [4]. This temporal correlation contains the user's habit of submitting jobs in the cluster. We autoregressively analyze this temporal correlation between jobs to obtain the possible job attribute distribution. Assume a specific user u , we need to generate job attributes $\{A(j_1^u), A(j_2^u), \dots\}$ for each job in u 's simulated job sequence $\{j_1^u, j_2^u, \dots\}$. The basis of this generation process is an autoregressive analysis on u 's historical job attribute sequence, $\{A(j_{h1}^u), A(j_{h2}^u), A(j_{h3}^u), \dots\}$.

Generally, for the attribute generation of user u 's i -th job, we have to extract information from the previous part of the historical sequence to obtain its attribute distribution:

$$M(j_i^u) = \text{softmax}[\text{extrac}\{A(j_{h(i-n)}^u), \dots, A(j_{h(i-2)}^u), A(j_{h(i-1)}^u)\}], \quad (4)$$

where $M(j_i^u)$ represents the multinomial distribution of attribute values of user u 's i -th job. In the following section, we will introduce how the approach proposed in this paper autoregressively extracts historical sequence information to perform workload generation.

Attentive Sequential Generation. The proposed approach mainly uses the attention layer to extract historical sequence information. Compared with RNNs (Recurrent neural networks) and statistical time series models such as ARIMA (Autoregressive Integrated Moving Average Model), Transformer networks based on the attention mechanism are superior in extracting long-term dependencies in sequence information [31]. The attention mechanism can be defined as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}, \quad (5)$$

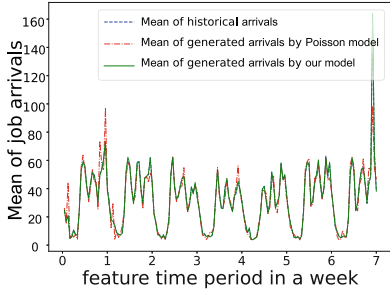
where \mathbf{Q} represents the queries, \mathbf{K} the keys and \mathbf{V} the values. The attention block calculates a weighted sum of all values, where the weight between query i and value j relates to the interaction between query i and key j . The scale factor \sqrt{d} is to avoid overly large values of the inner product. In our method, the attention layer is used to extract historical sequence information:

$$\begin{aligned} \mathbf{A}_i^u &= \{A(j_{h(i-n)}^u), \dots, A(j_{h(i-2)}^u), A(j_{h(i-1)}^u)\}, \\ \mathbf{E}_i^u &= \text{Attention}(\mathbf{A}_i^u \mathbf{W}^Q, \mathbf{A}_i^u \mathbf{W}^K, \mathbf{A}_i^u \mathbf{W}^V), \end{aligned} \quad (6)$$

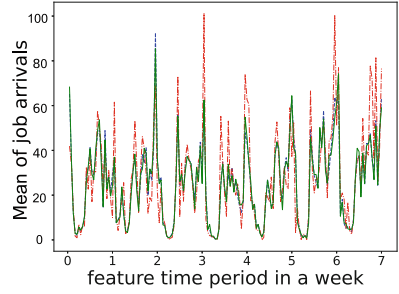
where $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d \times d}$ is the linear projection matrices. The intermediate representation \mathbf{E}_i^u is then transformed through the feed-forward layer and softmax layer, and finally the multinomial distribution $M(j_i^u)$ of the attributes of job j_i^u is obtained. $M(j_i^u) \in \mathbb{R}^{k \times D}$ is the probability distribution of job attributes based on the user's long-term habits obtained by autoregressively analyzing the user u 's historical job sequence. $A(j_{h(i)}^u) \in \mathbb{R}^{k \times D}$ before embedding is the one-hot code of the job attributes actually submitted by the user in this order, which reflects the short-term considerations of user u . Generally, we randomly generate job attributes for all jobs in the simulated job sequence according to the probability distribution $[M(j_i^u) + A(j_{h(i)}^u)]/2$. At this point, the workload generation that mimics the cluster scheduling behavior but is not restricted to historical workload patterns is completed.

4 Evaluation

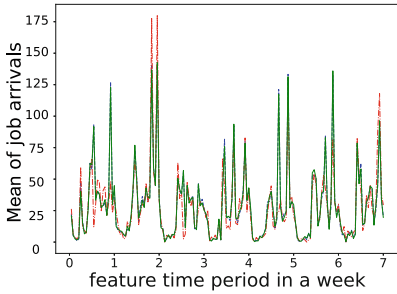
In this section, we evaluate whether synthetic workloads can mimic the scheduling behavior exhibited by real workloads from multiple perspectives. We first evaluate the generated job arrivals and then evaluate the overall workload generation results. Also, we evaluate our workload generation model on a use case.



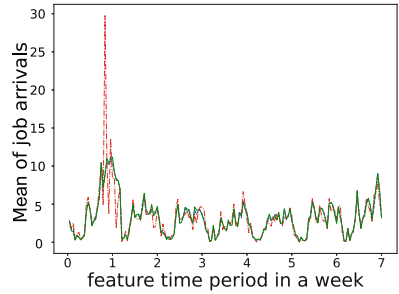
(a) Platform A



(b) Platform B



(c) Platform C



(d) Platform D

Fig. 2. Evaluation of job arrival models with metric *Mean*.

4.1 Evaluation of Job Arrival Generation

We evaluate our job arrival generation model by comparing its outputs with the outputs of Poisson regression model as the baseline. We use the following evaluation metrics:

- *Mean*, is the average number of job arrivals in each feature interval.
- *Devi*, is the regression deviance that is twice the difference between the maximum achievable log-likelihood and the log-likelihood of the fitted model: $Devi = 2[l(\mathbf{y}) - l(\boldsymbol{\alpha}, \boldsymbol{\mu})]$.

As described in Sect. 3.1, the feature vector of each interval in the job arrival model, which is a 168-dimensional one-hot code (7 days a week, multiplied by 24 h a day), represents the one-hour interval in one hour of the day of the week.

We evaluate job arrival generation results over time intervals represented by all 168 different one-hot feature vectors. The evaluation will compare whether the distribution of simulated job arrivals at each feature interval can approach the distribution of real job arrivals from the above metrics.

For the metric, *Mean*, it is intended to judge whether the arrival model can generate a reasonable number of job arrivals. The evaluation results for this metric are shown in Fig. 2. It can be seen that our model and the Poisson regression model both can generate a reasonable number of job arrivals in each feature interval. In each feature interval, the average number of generated job arrivals is very close to the statistical results in the real trace.

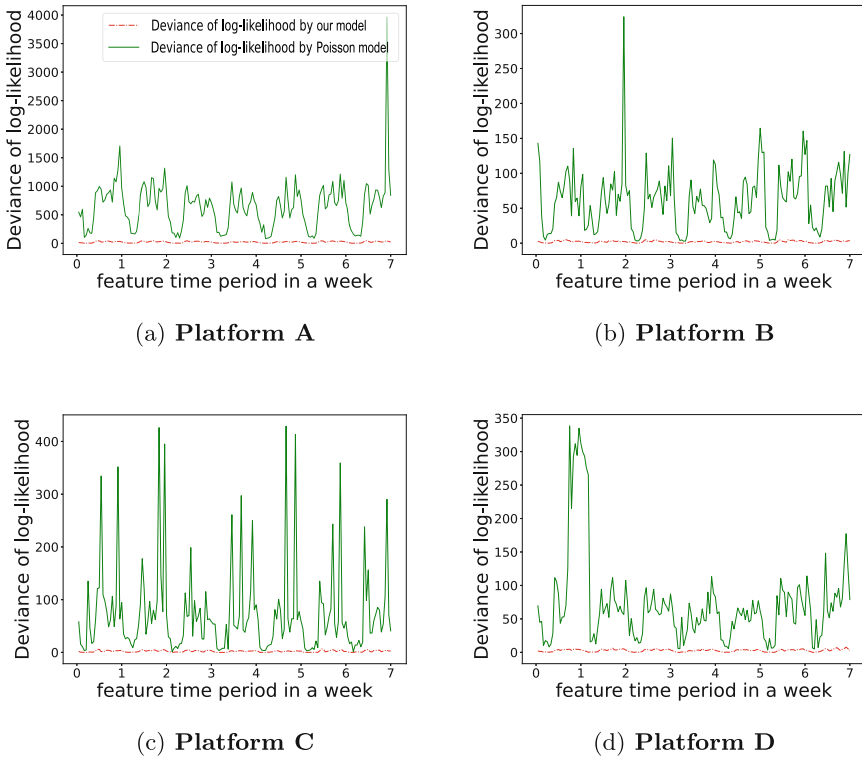


Fig. 3. Evaluation of job arrival models with metric *Devi*.

The metric, *Devi*, is intended to judge whether the regressions in arrival models converge well or not. The distribution obtained by Poisson regression will be approximated for calculation of deviance as a negative binomial distribution of $\mu = \lambda$ and $\alpha = 10^{-5}$. The evaluation results for this metric are shown in Fig. 3. Taking platform A as an example, it can be seen that naive Poisson regression is difficult to converge on the trace of platform A, while our model converges

perfectly. This result shows that although the Poisson regression model can generate a reasonable number of job arrivals, it cannot restore the overdispersion in the original data distribution. Notably, the periodic variation of deviance in Poisson regression is strongly correlated with the periodic variation of the mean of job arrivals in all features. The periodic variation in the mean of job arrivals is due to the change in the amount of user job submissions caused by the alternation of day and night. Similar periodic variation in regression deviance of job arrivals indicates that there is a strong positive correlation between the degree of overdispersion of job arrivals and the number of job arrivals in the evaluated HPC cluster.

4.2 Evaluation of Workload Generation

We evaluate our overall workload generation results by comparing them with the results of existing methods as baselines:

- **Multinomial:** This type of method directly generates workloads based on the multinomial distribution of user job attributes based on statistics on all historical traces. Many classic HPC workload modeling methods are of this type [8, 25, 35].
- **RepeatFlav:** This method generates duplicate job attributes for all user jobs in a period based on the most frequently occurring user job attributes in this period. It was used as the main baseline in the work of Bergsma et al. [4].
- **Bergsma:** The work of Bergsma et al. serves as our main baseline [4]. Their method can represent the state-of-the-art deep learning-based workload generation methods. They used LSTM combined with survival prediction as the backbone network to conduct autoregressive analysis of historical sequence information and gradually job attribute distribution step by step.

We use the following evaluation metrics:

- **Coverage Accuracy**, indicates how accurately the distribution used for job attribute generation covers the real job attributes. For a job, if the probability of generating real job attribute values in its generation distribution exceeds 80%, then this step of generation is counted as an accurate generation.
- **Cosine Similarity**, represents the cosine similarity between synthetic workload sequences and real workload sequences, $\text{Cosine Similarity}(\mathbf{W}, \hat{\mathbf{W}}) = \mathbf{W} \cdot \hat{\mathbf{W}} / (\|\mathbf{W}\| \|\hat{\mathbf{W}}\|)$

These two metrics measure the correlation between model output and real workloads from different perspectives.

Table 2 shows the metrics comparison on all platform traces of job requested resource sequence generated by our model and baselines. It can be seen that

Table 2. Evaluation of job requested resource generation.

Platform A		
	Coverage Accuracy	Cosine Similarity
Multinomial	67.15%	0.5018
RepeatFlav	72.17%	0.5183
Bergsma	80.28%	0.6672
Our model	82.93%	0.7042
Platform B		
	Coverage Accuracy	Cosine Similarity
Multinomial	77.93%	0.6512
RepeatFlav	80.82%	0.6791
Bergsma	88.21%	0.7117
Our model	89.14%	0.7291
Platform C		
	Coverage Accuracy	Cosine Similarity
Multinomial	69.41%	0.5293
RepeatFlav	78.96%	0.6174
Bergsma	83.22%	0.6713
Our model	83.58%	0.7132
Platform D		
	Coverage Accuracy	Cosine Similarity
Multinomial	77.37%	0.6884
RepeatFlav	79.51%	0.7375
Bergsma	85.16%	0.7927
Our model	90.84%	0.9015

our model has higher coverage accuracy and can better reflect the job requested resource correlation in real workloads, compared to all baselines. In all platforms on the dataset, the variation of job requested resources is not large, so even Multinomial model can effectively generate the job requested resource for all jobs.

Table 3 is the metrics comparison on all platform traces of job run time generated by our model and baselines. Unlike the evaluation of the job requested resource model, the performance of Multinomial model and RepeatFlav model are poor, due to the variation of run time being too large in all platform traces. In contrast, Bergsma model and our model achieve much better results on the generation of job run time. Compared with the Bergsma model, our job running time generation method also has advantages in both metrics.

Table 3. Evaluation of job run time generation.

Platform A		
	Coverage Accuracy	Cosine Similarity
Multinomial	27.31%	0.2018
RepeatFlav	37.26%	0.1475
Bergsma	51.40%	0.6034
Our model	57.32%	0.6437
Platform B		
	Coverage Accuracy	Cosine Similarity
Multinomial	34.76%	0.2910
RepeatFlav	45.81%	0.3049
Bergsma	60.84%	0.6826
Our model	60.43%	0.6728
Platform C		
	Coverage Accuracy	Cosine Similarity
Multinomial	33.19%	0.2941
RepeatFlav	44.29%	0.3121
Bergsma	59.11%	0.6186
Our model	59.41%	0.6472
Platform D		
	Coverage Accuracy	Cosine Similarity
Multinomial	43.72%	0.3219
RepeatFlav	50.14%	0.4112
Bergsma	69.11%	0.7236
Our model	69.17%	0.7311

4.3 Use Case of Workload Generation

We evaluate our workload generation model on a real-world use case: cluster planning [4]. Cluster planning is when the workload in the cluster is overloaded/underloaded, we need to plan the appropriate number of nodes to add/reduce. For this problem, synthetic workloads are necessary to more fully simulate and verify whether the cluster is load balanced after adding or reducing the corresponding number of nodes. Figure 4 shows our evaluation results on the trace of platform A of whether the synthetic workloads reflect cluster load levels. In this experiment, we assume that the cluster has no limit on the number of computing resources, and then count the number of computing node cores occupied by the workloads in each time period to represent the cluster load level. It can be seen that our synthetic workloads can well reflect the real load level, where we treat the attribute distribution of each job as a discretization of the

Gaussian distribution to calculate the 90% confidence interval of the synthetic load level. Our synthetic workloads can be generated in large numbers based on random patterns while simulating real load levels, making cluster planning simulations more sufficient and credible.

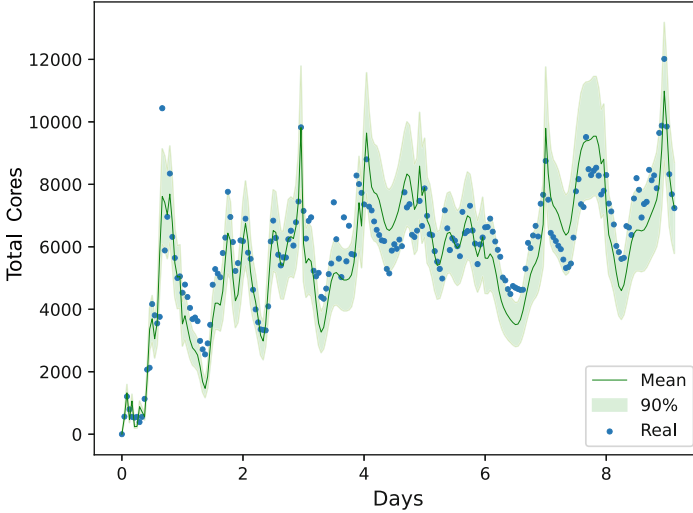


Fig. 4. Evaluation of whether the synthetic workloads reflect cluster load levels.

5 Related Work

5.1 Workload Modeling in HPC

Workload modeling has long been an active topic in the HPC community. Many existing works [8, 25, 35] were devoted to using statistical multinomial distribution to fit the workload modeling of supercomputing systems, which is a classic HPC modeling method. Rodrigo et al. further characterized the workloads in their HPC systems based on the system life cycle and evolution trends [29] and proposed a modeling method for heterogeneous workloads based on this portrayal [28]. Our approach is inspired by these existing state-of-the-art methods and further proposes a more comprehensive workload modeling for workload generation.

5.2 Workload Modeling in Cloud Computing

For cloud computing, although the main workload service types are different, their workload scheduling forms are similar to those in classic supercomputing systems. The mainstream method of workload modeling in cloud computing is to describe workloads' changing stochastic processes. Calheiros et al. introduced a

workload prediction model based on ARIMA [5]. Gao et al. conducted a detailed study on the prediction of cloud resource allocation [15]. Cortez et al. constructed a time series model of the complete workload trend from the historical characteristics of virtual resource workload sequences [9]. Bergsma et al. established workload modeling also based on historical sequence characteristics, which can effectively generate complex and realistic workloads [4]. Our approach also draws from stochastic considerations and generation techniques in modeling workloads in cloud computing.

5.3 Resource Management in HPC

Workload modeling is beneficial for managing resources to improve utilization of HPC clusters. Many resource management services are based on workload profiling. The foundation of performance modeling and optimization of existing scheduling systems such as Slurm is workload modeling [26,34]. Cluster planning also relies on workload models for more adequate simulation [4]. Workload modeling and generation is more widely used in the evaluation of scheduling algorithms [24,30], ranging from classic schedulers [17], heuristic-based schedulers [7,20,22], prediction-based schedulers [14], deep reinforcement learning-based schedulers [12,13,27,33,36,37]. Motivated by these existing works, we plan to design a novel cluster scheduler based on our workload modeling in future work.

6 Conclusion

In this paper, we proposed a method to model the workload in HPC clusters and generate synthetic workloads for HPC clusters. We combined statistical modeling and autoregressive analysis in our model to better characterize the workload scheduling behavior exhibited by real workloads. Using the proposed model, we can generate a synthetic workload at any time involved in modeling, which can reflect the most consistent workload distribution based on accurate historical characteristics extraction of real workloads. The proposed method greatly improved the quality of automatically generated workloads and made the workload generation model more reliable for resource management services, such as cluster planning and scheduling.

In the future, we plan to further evaluate how the synthetic workload reflects real scheduling behavior and determine whether it can play a role in data augmentation. On this basis, we will design a more efficient data-driven learning-based cluster scheduler that can be further enhanced with synthetic workloads for training.

Acknowledgements. We thank the anonymous reviewers for their valuable comments. This study was supported by NSF of China (Grant No. 62202441). The experiments of this study were supported by Supercomputing Center of University of Science and Technology of China.

References

1. Center for High Performance Computing in Shanghai Jiao Tong University. <https://hpc.sjtu.edu.cn/>
2. Shanghai Supercomputing Center. <https://www.ssc.net.cn/>
3. Supercomputing Center of University of Science and Technology of China. <https://scc.ustc.edu.cn/>
4. Bergsma, S., Zeyl, T., Senderovich, A., Beck, J.C.: Generating complex, realistic cloud workloads using recurrent neural networks. In: Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles, pp. 376–391 (2021)
5. Calheiros, R.N., Masoumi, E., Ranjan, R., Buyya, R.: Workload prediction using ARIMA model and its impact on cloud applications' QoS. *IEEE Trans. Cloud Comput.* **3**(4), 449–458 (2014)
6. Chapin, S.J., et al.: Benchmarks and standards for the evaluation of parallel job schedulers. In: Feitelson, D.G., Rudolph, L. (eds.) *JSSPP 1999*. LNCS, vol. 1659, pp. 67–90. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-47954-6_4
7. Chen, X., et al.: A WOA-based optimization approach for task scheduling in cloud computing systems. *IEEE Syst. J.* **14**(3), 3117–3128 (2020)
8. Cirne, W., Berman, F.: A comprehensive model of the supercomputer workload. In: Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization, *WWC-4* (Cat. No. 01EX538), pp. 140–148. IEEE (2001)
9. Cortez, E., Bonde, A., Muzio, A., Russinovich, M., Fontoura, M., Bianchini, R.: Resource central: understanding and predicting workloads for improved resource management in large cloud platforms. In: Proceedings of the 26th Symposium on Operating Systems Principles, pp. 153–167 (2017)
10. Curtiss, L.A., Redfern, P.C., Raghavachari, K.: Gaussian-4 theory. *J. Chem. Phys.* **126**(8), 084108 (2007)
11. Dickson, J., et al.: Enabling portable I/O analysis of commercially sensitive HPC applications through workload replication. In: Cray User Group 2017 Proceedings (CUG2017 Proceedings), pp. 1–14 (2017)
12. Fan, Y., Lan, Z.: DRAS-CQSim: a reinforcement learning based framework for HPC cluster scheduling. *Softw. Impacts* **8**, 100077 (2021)
13. Fan, Y., Lan, Z., Childers, T., Rich, P., Allcock, W., Papka, M.E.: Deep reinforcement agent for scheduling in HPC. In: 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 807–816. IEEE (2021)
14. Găinaru, A., Aupy, G.P., Sun, H., Raghavan, P.: Speculative scheduling for stochastic HPC applications. In: Proceedings of the 48th International Conference on Parallel Processing, pp. 1–10 (2019)
15. Gao, J., Wang, H., Shen, H.: Machine learning based workload prediction in cloud computing. In: 2020 29th International Conference on Computer Communications and Networks (ICCCN), pp. 1–9. IEEE (2020)
16. Ghaderi, J.: Randomized algorithms for scheduling VMs in the cloud. In: IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, pp. 1–9. IEEE (2016)
17. Gómez-Martín, C., Vega-Rodríguez, M.A., González-Sánchez, J.L.: Fattened back-filling: an improved strategy for job scheduling in parallel systems. *J. Parallel Distrib. Comput.* **97**, 69–77 (2016)
18. Hafner, J.: *Ab-initio* simulations of materials using VASP: density-functional theory and beyond. *J. Comput. Chem.* **29**(13), 2044–2078 (2008)

19. Heath, M.T.: *Scientific Computing: An Introductory Survey*, Revised Second Edition. SIAM (2018)
20. Houssein, E.H., Gad, A.G., Wazery, Y.M., Suganthan, P.N.: Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends. *Swarm Evol. Comput.* **62**, 100841 (2021)
21. Klein, N., Kneib, T., Lang, S.: Bayesian generalized additive models for location, scale, and shape for zero-inflated and overdispersed count data. *J. Am. Stat. Assoc.* **110**(509), 405–419 (2015)
22. Kuchumov, R., Korkhov, V.: Analytical and numerical evaluation of co-scheduling strategies and their application. *Computers* **10**(10), 122 (2021)
23. Lin, W., Yao, K., Zeng, L., Liu, F., Shan, C., Hong, X.: A GAN-based method for time-dependent cloud workload generation. *J. Parallel Distrib. Comput.* **168**, 33–44 (2022)
24. Liu, Y., Wang, L., Wang, X.V., Xu, X., Zhang, L.: Scheduling in cloud manufacturing: state-of-the-art and research challenges. *Int. J. Prod. Res.* **57**(15–16), 4854–4879 (2019)
25. Lublin, U., Feitelson, D.G.: The workload on parallel supercomputers: modeling the characteristics of rigid jobs. *J. Parallel Distrib. Comput.* **63**(11), 1105–1122 (2003)
26. Reuther, A., et al.: Scalable system scheduling for HPC and big data. *J. Parallel Distrib. Comput.* **111**, 76–92 (2018)
27. Reza, M.F., Zhao, B.: Deep reinforcement learning with different rewards for scheduling in high-performance computing systems. In: 2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), pp. 183–186. IEEE (2021)
28. Rodrigo, G.P., Östberg, P.O., Elmroth, E., Antypas, K., Gerber, R., Ramakrishnan, L.: Towards understanding HPC users and systems: a NERSC case study. *J. Parallel Distrib. Comput.* **111**, 206–221 (2018)
29. Rodrigo Álvarez, G.P., Östberg, P.O., Elmroth, E., Antypas, K., Gerber, R., Ramakrishnan, L.: HPC system lifetime story: workload characterization and evolutionary analyses on NERSC systems. In: Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing, pp. 57–60 (2015)
30. Singh, S., Chana, I.: A survey on resource scheduling in cloud computing: issues and challenges. *J. Grid Comput.* **14**(2), 217–264 (2016). <https://doi.org/10.1007/s10723-015-9359-2>
31. Vaswani, A., et al.: Attention is all you need. In: *Advances in Neural Information Processing Systems*, vol. 30 (2017)
32. Ver Hoef, J.M., Boveng, P.L.: Quasi-Poisson vs. negative binomial regression: how should we model overdispersed count data? *Ecology* **88**(11), 2766–2772 (2007)
33. Wang, Q., Zhang, H., Qu, C., Shen, Y., Liu, X., Li, J.: RLSchert: an HPC job scheduler using deep reinforcement learning and remaining time prediction. *Appl. Sci.* **11**(20), 9448 (2021)
34. Yoo, A.B., Jette, M.A., Grondona, M.: SLURM: simple Linux utility for resource management. In: Feitelson, D., Rudolph, L., Schwiegelshohn, U. (eds.) *JSSPP 2003*. LNCS, vol. 2862, pp. 44–60. Springer, Heidelberg (2003). https://doi.org/10.1007/10968987_3
35. You, H., Zhang, H.: Comprehensive workload analysis and modeling of a petascale supercomputer. In: Cirne, W., Desai, N., Frachtenberg, E., Schwiegelshohn, U. (eds.) *JSSPP 2012*. LNCS, vol. 7698, pp. 253–271. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35867-8_14

36. Zhang, D., Dai, D., He, Y., Bao, F.S., Xie, B.: RLScheduler: an automated HPC batch job scheduler using reinforcement learning. In: SC 2020: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–15. IEEE (2020)
37. Zhao, J., Rodríguez, M.A., Buyya, R.: A deep reinforcement learning approach to resource management in hybrid clouds harnessing renewable energy and task scheduling. In: 2021 IEEE 14th International Conference on Cloud Computing (CLOUD), pp. 240–249. IEEE (2021)