Queue congestion prediction for large-scale high performance computing systems using a hidden Markov model

Ju-Won Park¹ · Min-Woo Kwon¹ · Taeyoung Hong¹

Accepted: 3 February 2022 / Published online: 28 February 2022
© The Author(s) 2022

Abstract

To share limited, large-capacity resources, the high-performance computing field provides services by allocating available resources to jobs through batch job schedulers. Therefore, it is natural that a queue waiting time occurs until the resources are available if resources are not sufficient. The prediction of queue waiting time is very useful to improve overall resource utilization. However, the queue waiting time is very difficult to predict because it is significantly affected by the many factors such as applied scheduling algorithm and characteristics of the executed job. In this study, a method of predicting queue waiting time using only the historical log data created by the batch job scheduler is examined. Specifically, a method of predicting queue waiting time based on a hidden Markov model is proposed. It has the following three stages. First, outliers are removed by applying the outlier detection algorithm using a statistics-based parametric method. Second, the parameters of the hidden state are estimated using the observed queue waiting time sequence based on the historical job log. Third, the queue waiting interval at time $t + 1$ is provided using the estimated parameters at time t . Comparing the prediction accuracy with those of the other prediction methods, experimental results show that the proposed algorithm improves the prediction accuracy by up to 60%.

Keywords Queue waiting time · Hidden Markov model · Log analysis

✉ Ju-Won Park
juwon.park@kisti.re.kr

Min-Woo Kwon
mwkwon81@kisti.re.kr

Taeyoung Hong
tyhong@kisti.re.kr

¹ Korea Institute of Science and Technology Information, 245 Daehak-ro, Daejeon 34141, Republic of Korea

1 Introduction

Recently, the demand for large-capacity cluster systems has soared owing to the requirement of significant computing resources in basic science fields that traditionally use considerable computing resources such as meteorology, astronomy, and space science, as well as in computer science fields such as artificial intelligence (AI) and computer vision. To share limited large-capacity resources with a large number of users efficiently, the high-performance computing (HPC) field provides services by allocating available resources to jobs through batch job schedulers, such as Slurm [1] and PBS [2]. The user creates a script file, including an executable file and input parameters needed for the execution of his/her application program in a certain format based on the job scheduler and submits it to the batch job scheduler. The batch job scheduler matches the user-requested resources and available resources and allocates available resources sequentially according to a certain service policy, involving first in first out (FIFO) [3], fairness-aware [4], or energy-efficient [5]. Thereafter, the user job is executed using the allocated computing resources. If the available resources are not sufficient, the user job waits in a queue until the currently running job is finished and available resources are secured, resulting in a queue waiting time.

There are several advantages to predicting these queue waiting times in advance. First, from the user's point of view, a user job can be submitted by selecting an appropriate queue according to the deadline for the job. This is very useful because it is necessary to manage the deadline of each job in the HPC environment where many large workload jobs are executed. Second, in diverse environments, such as grids where many clusters are distributed and used, a user can submit a job by selecting the shortest queue waiting time in terms of the predicted queue waiting time. This leads to jobs being distributed across the network, increasing overall resource utilization.

However, queue waiting times are very difficult to predict because they are significantly affected by the applied scheduling algorithm, characteristics of the executed job, and scheduling policies of the cluster operating organization. This is especially difficult when the waiting times are predicted based on the historical job log alone without additional relevant information, such as the applied scheduling algorithm. In this paper, a method of predicting queue waiting time using only the historical log data created by the batch job scheduler is examined. To this end, first, the concept of queue congestion, which refers to the degree of congestion in the queue waiting time aspect, is introduced. The higher the congestion, the longer is the queue waiting time in the future, and the lower the congestion, the shorter is the queue waiting time in the future. Furthermore, a method of predicting queue waiting time range based on a hidden Markov model is proposed. The proposed algorithm has the following three stages. First, outliers are removed by applying the outlier detection algorithm using a statistics-based parametric method. Second, the parameters of the hidden state, state transition probability and emission probability are estimated via the Baum–Welch algorithm [6] using the queue waiting time sequence obtained based on the historical job log.

Third, the queue waiting interval at time $t + 1$ is provided using the state transition probability and emission probability estimated at time t .

The remainder of this paper is organized as follows. Section 2 introduces the batch job scheduler log and related studies. In Sect. 3, the applicability of the conventional feature-based prediction method and the time-series-based prediction method are described. Section 4 introduces the concept of queue congestion and provides a detailed description of the proposed algorithm. Furthermore, a system based on the proposed concept is implemented. In Sect. 5, the proposed method is validated by comparing it against the conventional feature-based analysis method and univariate analysis method. Finally, conclusions are presented in Sect. 6.

2 Background

2.1 Dataset

The National Supercomputing Center at the Korea Institute of Science and Technology Information (KISTI) provides supercomputing services to support the national research. The center was first established with a supercomputer, Cray-25 in 1988; as of 2018, the fifth supercomputer has been introduced, providing HPC resources to the national researchers. The fifth supercomputer named *Nurion* is Cray's CS500 model, which was built with a system having a many-core central processing unit (CPU) of 68 cores per socket, in parallel with Intel Xeon CPU nodes. It consists of approximately 8,300 nodes and is equipped with high-performance interconnect and Burst Buffer technology that can smoothly process large I/O requests. PBS Pro is used as a batch job scheduler for user job and computing resource management. Log files of PBS Pro scheduler are recorded in the following format.

< date and time > ; < record type > ; < ID string > ; < message text >

In the record type field, codes, such as A (job was aborted by the server), D (job was deleted by request), and E (job was terminated), are recorded according to the type of the recorded log. The ID string has the record of the job ID, and the message text has the record of job execution information in a text format of “key = value” according to record type. In this paper, the cases where the server or user canceled the job are excluded - i.e., by filtering only logs with a record type of E. Logs with the record type of E contain 22 features, including the time at which the job was created (ctime) and the time that the job entered the queue (qtime) [7].

2.2 Related work

Previous research in this field can be categorized into two types. The first type involves the feature-based prediction approach that derives predictions from similar observations in the historical data [8–10]. In [9], the authors proposed an Instance-Based Learning (IBL) technique to predict response time, which is

composed of application running and queue wait times by mining historical workloads. To seek the similarity, job attributes (e.g., group name, user name, number of CPUs, etc.), policy attributes (e.g., group name, user name, queue name), and resource state (e.g., number of free CPUs, categorized number of running jobs and number of queue jobs, etc.) are used. In [8], the authors had developed a framework for predicting the range of queue waiting times for jobs by employing multi-class classification of similar jobs in history. First, they predicted the point wait time of the job using the dynamic k-Nearest Neighbor (kNN) method. Subsequently, they performed a multi-class classification using support vector machines (SVMs) among all the classes of the jobs. In [10], an on-line waiting time prediction method is presented. The prediction technique largely comprises three phases. The first phase is the pre-processing of data in constant time intervals. In the second phase, major features are selected through a factor analysis and clustering is conducted based on the selected features. In the third phase, the waiting time of the next job is predicted using the sliding window method based on the jobs that were clustered.

The second type of previous work comprises the univariate approach using only the queue waiting time sequence. Downey [11, 12] explored a log-uniform distribution to model the remaining lifetimes of the jobs. He showed that it is possible to predict the queue waiting time with accuracy if the scheduling algorithm is known. Similarly, in [13] they performed the queue waiting time prediction using four workloads and three scheduling algorithms. In this work, the greedy and genetic algorithm was used to dynamically determine fit-similarity so that the accuracy of prediction could be improved. In [14], the authors had developed a method, the binomial method batch-queue predictor (BMBP), which was shown to make accurate and correct predictions for bounds on job wait times. The BMBP predicts quantiles directly from the historical data. It was implemented as a trace-based simulation that takes as inputs, a historical job trace, a user-specified quantile, and a confidence level. In [15], the authors described one parametric model fitting technique and two non-parametric prediction techniques, comparing their accuracy in predicting the quantiles of empirically observed machine availability distributions. To estimate the parameters from the observed data, a maximum likelihood estimation (MLE) technique was used. In addition, the resample and binomial methods were used to calculate the confidence bound for the quantile of the random variable model based on the observation data. In [16], the authors introduced an online system, QBETS for predicting batch queue delay. This system is composed of four novel and interacting components: a predictor based on nonparametric inference, an automated change point detector, a model-based clustering of jobs having similar characteristics, and an automatic downtime detector to identify systemic failures that affect job queuing delay. It predicted the bounds (with specific levels of certainty) on the amount of queue delay each individual job would experience.

Although some studies such as [9, 11–13] focused on point-value predictions for job wait times in a batch queue, many studies showed that the determination of an exact value is practically impossible because of the complex and highly skewed nature of wait-time data [14, 17, 18]. Therefore, in this paper, the time range is predicted where the value is most likely to exist instead of point-value predictions. For

this, the hidden Markov model is used. To the best of our knowledge, this is the first application of the model to large-scale parallel workload queue waiting time prediction, although it has been used in many fields such as signal detection, speech recognition, mobile computing, and disease forecasting [19–22].

3 Statistical job analysis in supercomputer

In this section, the relationships between the statistical information of queue waiting times and each job feature is examined based on the logs of jobs executed on *Nurion* for six months from February 2019 to July 2019. In the case of PBS Pro, the logs does not have records of the queue waiting time information. However, the queue waiting time can be calculated based on the difference between the time at which the job was submitted (qtime) and the time at which the job was executed (start).

3.1 Statistics of job scheduler log

Table 1 presents the statistics of jobs based on queue. The total number of jobs executed over the six months was 444,754, the total job execution time was 734,753,406 seconds, and the mean execution time per job was 16,521.41 seconds. According to the records in the log, *Nurion* operated by creating seven queues, and the number of jobs executed in each queue, mean execution time, and deviation are shown in Table 1. For 85% of all jobs, the service was provided through the normal queue, and the mean execution time of the jobs executed in the normal queue was 13,358 seconds with a deviation of 35,177 seconds. In the execution time aspect of the jobs executed in each queue, the long queue showed 78,698 seconds, which was the longest and 5.8 times longer than that of the normal queue.

Figure 1 shows a graph of the queue waiting time for all jobs, and Table 2 shows the statistical information. As can be seen in the figure and table, the queue waiting time is very widely distributed from 0 to 2,113,174 seconds, with a VAR value of 106,976 hours and SD value of 19,624.37 seconds, indicating a very large deviation.

Table 1 Statistics of job based on queues

Queue name	#of job	Total of run time (hr)	Avg. of run time (sec)	SD
Burst buffer	1915	5862.2	11,020	24,278
Commercial	12,292	6,0932	17,845	24,278
Exclusive	9540	56,886.7	21,467	47,858
Flat	11,883	62,042.6	18,796	42,824
Long	11,025	241,011.6	78,698	119,033
Norm skl	16,274	196,759.1	43,525	59,582
Normal	381,825	1,416,752.7	13,358	35,177

Fig. 1 Queue waiting time

Table 2 Statistics of the queue wait time

Min.	1st Qu.	median	mean	3rd Qu. (hr.)	Max.	VAR (hr.)	SD
0	3	8	3,328	15	586.9	106,976	19,624.37

Furthermore, after submitting jobs, 75% of all jobs were executed within 15 seconds, indicating that most jobs were executed without long waiting times; however, the mean queue waiting time of the remaining 15% of jobs was very long: 13,620.37 seconds.

3.2 Correlation between wait time and job feature

To predict the queue waiting time using job similarity, there must be some correlations between job features and queue waiting time. In this section, the correlations between the queue waiting time and the features of the jobs recorded in the PBS Pro logs to confirm the applicability of the feature-based algorithm is examined.

Queue waiting time can be determined when the job is not terminated but started. Therefore, to predict it, an analysis must be performed based only on the features that can be examined before the job is started. Thus, the queue waiting time must be predicted using the features shown in Table 3—those that can be identified before the job is started among the features recorded in the job scheduler log.

Queue waiting time, an independent variable predicted in this paper, is a continuous value; continuous values and categorical values are mixed in features, which are dependent variables. First of all, let us examine the correlations between the queue

Table 3 Available features for predictions

Attribute	Description	Value type
User	The user name under which the job submitted	Categorical
Group	The group name under which the job submitted	Categorical
Queue	The name of the queue which the job submitted	Categorical
Resource_list.mpiexecs	The number of processes requested by the job	Continuous
Resource_list.ncpus	The number of cpu requested by the job	Continuous
Resource_list.nodect	The number of nodes requested by the job	Continuous

waiting time and continuous dependent variables, such as resource_list.mpiexecs, resource_list.ncpus, and resource_list_nodect. For this, the Pearson correlation coefficient is used since it is the most popular method showing the correlation of two random continuous variables.

Let x and y be two zero-mean real-valued random variables. The Pearson correlation coefficient is defined as [23, 24]

$$\rho_{X,Y} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} \quad (1)$$

$$= \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} \quad (2)$$

$$= \frac{\sum_1^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_1^n (X_i - \bar{X})^2} \sqrt{\sum_1^n (Y_i - \bar{Y})^2}} \quad (3)$$

where $\bar{X} = \frac{1}{n} \sum_{i=1}^n x_i$ and $\bar{Y} = \frac{1}{n} \sum_{i=1}^n y_i$ denote the mean of x and y , respectively. The range of correlation coefficient is $-1 \leq \rho_{X,Y} \leq 1$, and a value near to -1 or 1 indicate a significant correlation. Table 4 shows the Pearson correlation coefficient's value between the amount of resource required—a numeric dependent variable - queue waiting time, and independent variables. As shown in Table 4, there is almost no correlation between the two variables. This contradicts the intuitive patterns that in general, “the greater the amount of resource required, the longer the queue waiting time”.

Table 4 Pearson correlation coefficient

Attribute	Pearson correlation coefficient
Resource_list.mpiexecs	0.06999193
Resource_list.ncpus	0.05065556
Resource_list.nodect	0.04914997

```
> bf.test(waitTime ~ queue, data = df)

Brown-Forsythe Test (alpha = 0.05)
-----
data : waitTime and queue

statistic : 1571.68
num df    : 6
denom df  : 38431.87
p.value   : 0

Result    : Difference is statistically significant.
-----
```

Fig. 2 Brown–Forsythe Test result between queue and queue waiting time

Fig. 3 Queue waiting time box plot according to the queue

Next, the correlation between queue waiting time and the categorical values - user, group, and queue, are examined. The Brown–Forsythe test method [25] is used for this, which checks if a variable is a major factor that distinguishes groups through the variance analysis of clustered groups; it is a typical technique for analyzing the relationship between a continuous and a categorical values. In the Brown–Forsythe test, the correlation between the queue waiting time values with users and groups is not identified. However, as shown in Fig. 2, a significant difference is presented within the queue values.

Therefore, the queue waiting time is examined more closely by clustering for each queue. Figure 3 below shows the distribution graph of the queue waiting time according to the queue in a box shape. The red marks show outliers, and the lower part, upper part, and solid line in the box show the first quartile values, third quartile values, and median values, respectively. Table 5 shows the number of jobs executed in each queue, mean queue waiting time, and the median queue waiting time. As

Table 5 Statistics of the queue waiting time based on queues

Queue name	Avg. of waiting time (sec)	Median	SD
Burst_buffer	2769	6	7,783
Commercial	6176	3	19,988
Exclusive	8858	13	23,074
Flat	4622	19	19,113
Long	16,819	9	54,160
Norm_skl	26,165	4,398	59,218
Normal	1697	8	12,058

shown in Fig. 3 and Table 5, more than 80% of all jobs in the flat queue and the normal queue are executed without a long waiting time: the recorded queue waiting time was less than 1 minute. The burst buffer, commercial, exclusive, and log queues also show a median value of less than 10 seconds, and more than 70% of all jobs are executed in 1 to 2 hours after entering the queue. However, in the case of normal_skl, the median queue waiting time is greater than 1 hour with a mean value of 7 hours or longer, indicating that the waiting time is very long. Furthermore, the number of executed jobs in normal_skl queue is the highest except for the normal queue. Therefore, if the queue waiting time of the normal_skl queue is estimated in advance and provided to the user, it can significantly help in decision-making regarding the appropriate queue to send the job.

Now, let us take a closer look at the queue waiting time of the jobs executed in the normal_skl queue. Table 6 shows the Pearson correlation coefficients for only the jobs executed in the normal_skl queue. Compared to Table 4, it can be seen that the correlation has increased significantly overall. Furthermore, because the correlation is positive, it can be seen that as the required number of processes and nodes increases, the waiting time increases. This is consistent with the general hypothesis. However, considering that the correlation coefficient is between 0.36 and 0.39, the correlation is not enough [26].

3.3 Stationary property of queue waiting time

As demonstrated in Sect. 3.2, considering that the correlation between the features and the queue waiting time is not high, it is very difficult to predict the queue waiting time based on the job similarity. In this section, using the fact that queue waiting

Table 6 Pearson correlation coefficient between the amount of resource required and wait time in normal_skl queue

Attribute	Pearson correlation coefficient
Resource_list.mpiproc	0.369
Resource_list.ncpus	0.353
Resource_list.nodect	0.395

time measurement values are univariate data, the applicability of the time-series analysis method is examined.

The observation data must have the stationary property if the time-series analysis is to be performed. The Kwiatkowski–Phillips–Schmidt–Shin Test (KPSS) method suggested by [27] is used to check if the analyzed data have the stationary property. As shown in Eq. 4, the KPSS assumes a model in which time-series data consist of the sum of deterministic trend, random walk, and a stationary error, as given in the equation below, based on a test method in which the null hypothesis is “an observable series is stationary”.

$$y_t = \xi_t + r_t + \epsilon_t \quad (4)$$

As shown in Fig. 4, since the p value is less than 0.05 in both the *level* and the *trend* test results, the null hypothesis is rejected. Equivalently, the observation data do not have the stationary property.

4 Batch queue state analysis using hidden Markov model

As analyzed in Sect. 3, considering that queue waiting time has a low correlation with the features and shows abnormal time-series characteristics, the feature-based prediction and univariate-based prediction are both not suitable for the prediction of queue waiting time. Because of this difficulty, many studies [14, 17] did not predict the exact point value of queue waiting time but provided bound or prediction range information that had a certain probability value. In similar fashion, this paper, the method of estimating the interval in which queue waiting time exists, such as less than 1 minute or between 1 minute and 30 minutes, rather than the exact point value of queue waiting time is discussed. In practice, the queue waiting interval information can be very useful when selecting an appropriate queue for one's job in an environment where multiple queues exist. To this end,

```
> tseries::kpss.test(target_data$waitTime, null = "Level")

KPSS Test for Level Stationarity

data: target_data$waitTime
KPSS Level = 5.347, Truncation lag parameter = 13, p-value = 0.01
```

(a) level result

```
> tseries::kpss.test(target_data$waitTime, null = "Trend")

KPSS Test for Trend Stationarity

data: target_data$waitTime
KPSS Trend = 4.6834, Truncation lag parameter = 13, p-value = 0.01
```

(b) trend result

Fig. 4 KPSS test results of the queue waiting time

this section introduces the concept of queue congestion and proposes a method of predicting the queue waiting range using a hidden Markov model.

4.1 Hidden Markov model-based queue congestion prediction method

First, the concept of queue congestion is introduced. Queue congestion refers to the degree of congestion in terms of the queue waiting time: the greater the congestion, the longer the queue waiting time. It can be generally thought of as the number of jobs stacked in the queue, but to be precise, the correlation between the number of jobs waiting in the queue and the queue waiting time is very low. In fact, when the Pearson correlation coefficient between the number of jobs in the queue and queue waiting time based on the above data is calculated, the result was 0.309. Therefore, the queue congestion presented in this paper can be seen as a state estimated according to the degree of congestion for the queue waiting time expected at the time t . For this, a method that applies the hidden Markov model method is proposed to estimate the queue congestion at time t and suggest the queue waiting range at time $t + 1$.

The hidden Markov model (HMM) is a model proposed by Baum and Petrie in [28] and is based on the Markov chain. Markov chain is a discrete probability process with the Markov property of the assumption that the state at time t is affected by only the previous state ($t - 1$), and it can be expressed by Eq. 5.

$$P(q_t = a \mid q_1 \dots q_{t-1}) = P(q_t = a \mid q_{t-1}) \quad (5)$$

In the Markov chain, the transition probability matrix can be obtained through the transition probability of the state, and if this is used, the probability value of each output sequence can be calculated using the observation value according to the time. Markov chain is very useful for analyzing and predicting observable events. In practice, however, it is often impossible to directly observed events that are to be measured. Equivalently, it is a case where the queue waiting time sequence is observed, but there are no metrics that can directly measure the queue congestion. In this case, the hidden Markov model is very useful.

As shown in Table 7, the state of the observable event at time t is O_t , and the hidden state is Q_t . Q_t follows the Markov chain, a discrete probability process that has the Markov property. The probability of transitioning from each state to the state of time $t + 1$ refers to the probability that each a_{ij} will transition from state i to state j using the transition probability matrix, and it can be expressed by Eq. 6.

$$\sum_{j=1}^N a_{ij} = 1 \quad \forall i \quad (6)$$

In this paper, a method for predicting the queue congestion and queue waiting time based on the queue waiting time sequence obtained from the scheduler job log by applying the HMM is proposed. Equivalently, the HMM method is applied to obtain the queue congestion (Q_t)—the hidden state at time t —from the queue waiting time sequence (O), which is the observation data; based on this, the queue waiting

Table 7 The notation of a hidden Markov model

Element	Notation
Hidden Markov model	λ
Length of observation data	T
Number of states	N
Number of symbols per state	M
Observation sequence	$O = \{o_t, t = 1, 2, \dots, T\}$
Hidden state sequence	$Q = \{q_t, t = 1, 2, \dots, T\}$
Possible value of each state	$\{s_i, i = 1, 2, \dots, T\}$
Possible symbols per state	$\{v_k, k = 1, 2, \dots, M\}$
Transition matrix	$A = (a_{ij}), a_{ij} = P(q_t = S_j \mid q_{t-1} = S_i), i, j = 1, 2, \dots, N$
Emission matrix	$B = (b_{ik}), b_{ik} = P(o_t = v_k \mid q_t = S_i), i = 1, 2, \dots, N, k = 1, 2, \dots, M$

interval at time $t + 1$ is estimated. The proposed method consists of three main phases.

1. *Remove Outliers* The log data are refined by removing the statistics-based outliers. There are various methods of detecting and removing outliers, such as [29–31]. In this paper, QQ plot position-based outlier points are detected and removed using the statistics-based parametric method proposed by [30].
2. *Train HMM* The queue waiting time values of the outlier-removed historical data set are used as input values to train the HMM, based on which the transition probability matrix and emission probability are obtained. To this end, the Baum-Welch algorithm [32], which was proposed as a type of estimation maximization (EM), is used. First, let us examine the forward-backward algorithm to understand the Baum–Welch algorithm.

In the hidden Markov model, in which the state transition probability A and the emission probability B are given, the forward algorithm calculates the observation sequence probability by summing the probabilities of all inferable hidden state paths based on the measured observation sequence, and the probability of the hidden state at time t being j is found using Eq 7.

$$\alpha_t(j) = P(o_1, o_2, \dots, o_t, q_t = j \mid \lambda) \quad (7)$$

It can be expressed as a value obtained by multiplying the sum of probabilities of transitioning from all hidden states at time $t - 1$ to state j by the function of the likelihood that the observation value o_t will be observed from the hidden state j as shown in Eq. 8.

$$\alpha_t(j) = \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(o_t) \quad (8)$$

In contrast, the backward probability refers to the probability that the observation data from $t + 1$ to T will exist in the hidden state i of time t ; it can be

obtained by Eq. 9, which is the product of the function of the likelihood that o_{t+1} will be observed from every hidden state existing at $t + 1$.

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \quad 1 \leq i \leq N, 1 \leq t < T \quad (9)$$

However, when the transition matrix and the emission matrix are unknown, a process for estimating them is required. Equivalently, the optimal parameters $\theta^* = \{\pi^*, A^*, B^*\}$ are inferred from the observation set $O = \{o_1, o_2, \dots, o_T\}$, and for this purpose, the Baum-Welch algorithm is used. The expected values of the state transition probability and emission probability to be found can be expressed as follows.

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i} \quad (10)$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \sum_{k=1}^N \xi_t(i, k)} \quad (11)$$

$$\hat{b}_j(v_k) = \frac{\text{expected number of transition in } j \text{ nd observing symbol } v_k}{\text{expected number of times in } j} \quad (12)$$

$$= \frac{\sum_{t=1}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)} \quad (13)$$

where $\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{j=1}^N \alpha_t(j) \beta_t(j)}$ and $\gamma_t(j) = \frac{\alpha_t(j) \beta_t(j)}{P(O|\lambda)}$. Based on Eqs. 11 and 13,

parameters A and B can be estimated using the Baum-Welch algorithm.

3. *Predict queue waiting time* The estimated A and B are used to calculate the object prediction value at time $t + 1$. The queue congestion and the queue waiting time at time $t + 1$ can be calculated using Eqs. 14 and 15, respectively.

$$\max_{q_{t+1}} p(q_{t+1} | q_t) = \max_{q_{t+1}} A_{q_t, q_{t+1}} \quad (14)$$

$$\max_{o_{t+1}} p(o_{t+1} | q_{t+1}) = \max_{o_{t+1}} B_{q_{t+1}, o_{t+1}} \quad (15)$$

4.2 Method of implementing hidden Markov model-based queue congestion system

In this section, KISTI MIRINAE is introduced. It is a system that uses traffic light colors to display the queue congestion and shows the state transition probability and emission probability estimated using the Baum-Welch algorithm.

KISTI MIRINAE is an application program provided through a Web browser without installing the application program by using JavaScript. As shown in Fig. 5, the KISTI MIRINAE system is divided into three major parts. First, the top part shows the congestion state of each queue by classifying it into one of six-step colors - green, yellow, pale blue, blue, mild red, and red - in an array form of cards, along with the number of jobs currently running, the number of jobs in the queue, and the number of jobs that have been completed in each queue currently provided. The user can intuitively identify the degree of the queue congestion immediately through the color of the queue provided. For example, in the case of the green color, the queue's congestion is low. In contrast, in the case where the queue congestion level is displayed in red, the queue congestion is very high, and it is expected that if a job is submitted now, the queue waiting time will be very long. The lower part of the queue card shows the estimated range of the queue waiting time at time $t + 1$. The middle section shows two bar graphs for the number of jobs submitted by day of the week and hour, respectively. Considering that the number of submitted jobs can be identified by day of week and hour, the congestion of the queue can be roughly inferred. The bottom part of KISTI MIRINAE shows the transition probability matrix and the emission probability matrix estimated through the Baum-Welch algorithm, as described in Sect. 4.1.

5 Experimental results

This section describes the performance of the proposed prediction algorithm presented in Sect. 4.1. The performance is examined by comparing the results measured through the feature-based prediction methods and the univariate prediction methods described in related work in Sect. 2.2. As typical feature-based prediction methods, the recursive partitioning and regression trees (rpart) and XGBoost methods are used. Moreover, as univariate prediction methods, the simple moving average and the autoregressive integrated moving average (ARIMA) method are used. For the analysis data, the jobs only executed in normal_skl are used, as described in Sect. 3.

First, after sorting the jobs executed in normal_skl queue in ascending order based on the submission date, the outliers detected using the statistics-based parametric method are removed to refine the experimental data. Thereafter, the outlier-removed data set is indexed sequentially.

As presented in Table 8, the number of jobs executed in normal_skl queue is 14,759. The number of jobs detected as outliers is 3,923, and after excluding them, the remaining 10,836 jobs were used in the experimental dataset. The test set for measuring the accuracy contains the latest 3,251 jobs, which corresponds to 30% of the total number of jobs in the entire dataset.

Fig. 5 KISTI MIRINAE system dashboard

Table 9 shows the ranges of the queue waiting time. This comprises six divisions, as suggested in [8]; however, considering that more than 46% of the total in the log data - i.e., analysis target - are executed within 1 hour, the 1-hour range is classified into three classes by subdividing.

Figures 6, 7, 8 show the confusion matrixes of the results predicted by the five algorithms - rpart, XGBoost, moving average, ARIMA, and the proposed algorithm. In the case of the feature-based algorithms, XGBoost and rpart, the predicted values are concentrated in classes 4 and 6. This is because the number of job feature that can be used for prediction is six, which is very small, and the correlation with the queue waiting time is very low. Equivalently, even for jobs with the same feature values, the difference in the queue waiting time between each job is very large. However, in the case of the proposed algorithm, ARIMA, and moving average where only the queue waiting time sequence, the predicted values are widely distributed by class compared to those of the feature-based algorithms. Specifically, as presented in Table 9, the classes are values that segmented continuous time into six steps. Therefore, it can be inferred that the accuracy is higher when predicted as an adjacent class compared to a distant class. For example, suppose a class 3 job—the actual value - is not predicted as class 3, this would lead to an incorrect predicted value. Even so, it is a more useful value when it is predicted as class 2 or 4, an adjacent class, than when predicted as class 6. In the case of the proposed algorithm and ARIMA, it is confirmed through Figs. 7 and 8 that the percentage of predicting as an adjacent class is higher compared to the other algorithms although the inference values and predicted values are different.

Table 8 Statistics of the experimental dataset

Statistics	Value
Total number of job	14,759
The number of jobs detected as outliers	3923
The index of test set	From 7586 to 10,836

Table 9 Time ranges of each class

Class	Time ranges
1	Less than or equal to 1 min
2	1 min to 30 minutes
3	30 minutes to 1 hour
4	1 hour to 4 hours
5	4 hours to 8 hours
6	Greater than 8 hours

Table 10 shows the accuracy and no information rate (NIR) calculated based on the confusion matrix. The NIR is the largest proportion of the observed classes. As a simple example, suppose two classes exist, and class 1 occupies 80% of the total; if the predicted values are all class 1, the accuracy is 80%. Therefore, a hypothesis test is also conducted to evaluate whether the overall accuracy rate is greater than the rate of the largest class. As can be seen in Table 10, the accuracy of the univariate-based algorithms (i.e. moving average, ARIMA, and the proposed algorithm) is relatively higher than that of the feature-based algorithms (i.e. rpart and XGBoost). In the case of XGBoost and rpart, NIR and accuracy are very similar, and the p value is greater than 0.05, implying that the predicted value is not significant. In the case of moving average, similarly, the predicted values are not significant because the NIR and accuracy are very similar, although the p value is smaller than 0.05. However, the proposed algorithm and ARIMA have accuracy values of 0.3891 and 0.3088, respectively, which are higher than those obtained from the other algorithms. In particular, the proposed algorithm shows a performance improvement of 60% compared to XGBoost and rpart, 38% compared to the moving average method, and 26% compared to ARIMA.

Let us examine Cohen's kappa values as another metric of data for checking the predictive power of the classification algorithms. This kappa value, as mentioned by [33], is a good metric for measuring the predictive power of a classification algorithm. As can be seen in Table 10, the kappa values of the univariate-based algorithm are generally higher than those of the feature-based algorithms.

Finally, let us examine the predictive power of the proposed algorithm for each class. As can be seen in Table 11, in the case of class 6, the accuracy is approximately 0.7, which is relatively high, and the accuracy of class 3 is 0.541, which is relatively low. However, as can be seen in the confusion matrix table, in the case of

Fig. 6 Confusion matrixes of rpart and XGboost

class 1, the percentage predicted as class 1 or 2 is 68% of the total. Furthermore, in the case of class 3, the value of predicting as an adjacent class—class 2 or 4—is 65% of the total; furthermore, if class 3 is included, it increases to 76%.

Fig. 7 Confusion matrix of moving average and ARIMA

6 Conclusion

The queue waiting time is very difficult to predict because it is significantly affected by the many factors such as the scheduling algorithm and characteristics of the executed job. Furthermore, a low correlation with the features and abnormal time-series characteristics make matters worse. In this paper, the method of estimating

Fig. 8 Confusion matrix of the proposed algorithm

Table 10 Prediction accuracy

Algorithm	Accuracy	NIR	<i>p</i> value [ACC > NIR]	kappa
rpart	0.2378	0.2329	0.2593	0.0433
XGBoost	0.2359	0.2329	0.3456	0.0495
Moving average	0.2808	0.2329	$1.335e^{-10}$	0.1291
ARIMA	0.3088	0.2329	$< 2.2e^{-16}$	0.1655
The proposed algorithm	0.3891	0.2329	$< 2.2e^{-16}$	0.2491

Table 11 Prediction accuracy for each class

Class	Accuracy	Sensitivity (Recall)	Specificity
Class 1	0.588	0.252	0.923
Class 2	0.620	0.491	0.749
Class 3	0.541	0.110	0.971
Class 4	0.666	0.555	0.778
Class 5	0.606	0.260	0.952
Class 6	0.692	0.512	0.873

the certain range in which the queue waiting time exists rather than the exact point value of queue waiting time was discussed. For this, the concept of queue congestion which represents the degree of congestion in terms of the queue waiting time was introduced. Then, a method that applies the hidden Markov model method to

estimate the queue congestion at time t and suggest the queue waiting interval at time $t + 1$ was proposed. Using experiments, the accuracy of the univariate-based algorithms (i.e. moving average, ARIMA, and the proposed algorithm) is relatively higher than that of the feature-based algorithms (i.e. rpart and XGBoost). In particular, the proposed algorithm shows a performance improvement of 60% compared to XGBoost and rpart, 38% compared to the moving average method, and 26% compared to ARIMA.

Appendix A: List of abbreviations

See Table 12

Table 12 List of abbreviations

Abbreviation	Explanation
AI	Artificial Intelligence
HPC	High-Performance Computing
FIFO	First In First Out
IBL	Instance-Based Learning
kNN	k-Nearest Neighbor
SVMs	Support Vector Machines
BMBP	Binomial Method Batch Queue Predictor
MLE	Maximum Likelihood Estimation
QBETS	Queue bounds estimation from time series
VAR	Variance
SD	Standard Deviation
KPSS	Kwiatkowski–Phillips–Schmidt–Shin Test
HMM	Hidden Markov Model
QQ plot	Quantile-Quantile Plot
EM	Estimation Maximization
rpart	Recursive partitioning and regression trees
ARIMA	AutoRegressive Integrated Moving Average
NIR	No Information Rate

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Yoo AB, Jette MA, Grondona M (2003) Slurm: simple linux utility for resource management. In: Proc. of the Workshop on job scheduling strategies for parallel processing, Springer, pp 44–60
2. Henderson RL (1995) Job scheduling under the portable batch system. In: Proc. of the Workshop on Job Scheduling Strategies for Parallel Processing, Springer, pp 279–294
3. Qian J, Srisa-An W, Seth S, et al (2016) Exploiting Fifo Scheduler to Improve Parallel Garbage Collection Performance. In: Proc. of the 12th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, pp 109–121
4. Salami B, Noori H, Naghibzadeh M (2020) Fairness-aware energy efficient scheduling on heterogeneous multi-core processors. *IEEE Trans Comput* 70(1):72–82
5. Zhou S, Jin M, Du N (2020) Energy-efficient scheduling of a single batch processing machine with dynamic job arrival times. *Energy* 209(118):420
6. Baum LE, Petrie T, Soules G et al (1970) A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Ann Math Stat* 41(1):164–171
7. Technologies A (2021) Altair PBS Professional 2021.1 administrator's guide
8. Kumar R, Vadhiyar S (2014) Prediction of queue waiting times for metascheduling on parallel batch systems. In: Proc of the Workshop on Job Scheduling Strategies for Parallel Processing, Springer, pp 108–128
9. Li H, Groep D, Wolters L (2005) Efficient response time predictions by exploiting application and resource state similarities. In: Proc of the 6th IEEE/ACM International Workshop on Grid Computing, IEEE, pp 8
10. Park JW (2019) Queue Witing Time Prediction for Large-Scale High-Performance Computing System. In: Proc of the International Conference on High Performance Computing & Simulation, IEEE, pp 850–855
11. Downey AB (1997a) Predicting queue times on space-sharing parallel computers. In: Proc of the 11th International Parallel Processing Symposium, IEEE, pp 209–218
12. Downey AB (1997b) Using queue time predictions for processor allocation. In: Proc of the Workshop on Job Scheduling Strategies for Parallel Processing, Springer, pp 35–57
13. Smith W, Taylor V, Foster I (1999) Using run-time predictions to estimate queue wait times and improve scheduler performance. In: Proc of the Workshop on Job scheduling strategies for Parallel Processing, Springer, pp 202–219
14. Nurmi D, Mandal A, Brevik J, et al (2006) Evaluation of a Workflow Scheduler Using Integrated Performance Modelling and Batch Queue Wait Time Prediction. In: Proc. of the 2006 ACM/IEEE conference on Supercomputing, IEEE, pp 29–29
15. Brevik J, Nurmi D, Wolski R (2004) Automatic methods for predicting machine availability in desktop grid and peer-to-peer systems. In: Proc. of the International Symposium on Cluster Computing and the Grid, IEEE, pp 190–199
16. Nurmi D, Brevik J, Wolski R (2007) Qbets: queue bounds estimation from time series. In: Proc of the workshop on job scheduling strategies for parallel processing, Springer, pp 76–101
17. Sonmez O, Yigitbasi N, Iosup A, et al (2009) Trace-based evaluation of job runtime and queue wait time predictions in grids. In Proc of the 18th ACM international symposium on High performance distributed computing, ACM, pp 111–120
18. Olivares M, Musalem A, Yung D (2020) Balancing Agent Retention and Waiting Time in Service Platforms. In: Proc of the 21st ACM Conference on Economics and Computation, ACM, pp 295–313
19. Juang BH, Rabiner LR (1991) Hidden markov models for speech recognition. *Technometrics* 33(3):251–272
20. Mor B, Garhwal S, Kumar A (2020) A systematic review of hidden markov models and their applications. *Archiv Comput Methods Eng* 28(3):1429–1448
21. Li J, Wu B, Sun X, et al (2021) Causal Hidden Markov Model for Time Series Disease Forecasting. In: Proc of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, pp 12,105–12,114
22. Jazayeri F, Shahidinejad A, Ghobaei-Arani M (2021) A latency-aware and energy-efficient computation offloading in mobile fog computing: a hidden markov model-based approach. *The J Supercomput* 77(5):4887–4916

23. Lee Rodgers J, Nicewander WA (1988) Thirteen ways to look at the correlation coefficient. *The Am Stat* 42(1):59–66
24. Benesty J, Chen J, Huang Y et al (2009) Pearson correlation coefficient. *Noise Reduction in Speech Processing*, vol 2. Springer, Berlin Heidelberg, pp 1–4
25. Brown MB, Forsythe AB (1974) Robust tests for the equality of variances. *J Am Stat Assoc* 69(346):364–367
26. Schober P, Boer C, Schwarte LA (2018) Correlation coefficients: appropriate use and interpretation. *Anesth Anal* 126(5):1763–1768
27. Kwiatkowski D, Phillips PC, Schmidt P et al (1992) Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *J Econ* 54(1–3):159–178
28. Baum LE, Petrie T (1966) Statistical inference for probabilistic functions of finite state markov chains. *The Ann Math Stat* 37(6):1554–1563
29. Hodge V, Austin J (2004) A survey of outlier detection methodologies. *Artif Intell Rev* 22(2):85–126
30. Van der Loo MP (2010) Distribution based outlier detection in univariate data. *Statistics Netherlands*
31. Tang J, Chen Z, Fu AW et al (2007) Capabilities of outlier detection schemes in large datasets, framework and methodologies. *Knowl Inform Syst* 11(1):45–84
32. Baum LE et al (1972) An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities* 3(1):1–8
33. Ben-David A (2008) About the relationship between roc curves and cohen's kappa. *Eng Appl Artif Intell* 21(6):874–882

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.