Journal of Information Processing Vol.29 283–294 (Mar. 2021)

[DOI: 10.2197/ipsjjip.29.283]

Regular Paper

Power Prediction for Sustainable HPC

Shigeto Suzuki[1,a)] Michiko Hiraoka[2] Takashi Shiraishi[1] Enxhi Kreshpa[1]
Takuji Yamamoto[1] Hiroyuki Fukuda[1] Shuji Matsui[2] Masahide Fujisaki[2] Atsuya Uno[3]

Received: July 22, 2020, Accepted: November 2, 2020

Abstract: Exascale computers consume huge amounts of power and their variation over time makes system energy management important. Because of time lag in cooling-units operation, predictive control is desirable for effective power control. In this work, we report a state-of-the-art power prediction model. Conventional methods with topic model use the power of past job as a prediction based on the similarity of job information. The prediction, however, fails, if there is no correct data before. To resolve this, we developed a recurrent neural network model with variable network size, which detects features of power shape from its power history and enables precise prediction during job execution. By integrating these models into a single algorithm, the optimal model is automatically adopted for prediction according to the job status. We demonstrated high-precision prediction with an average relative error of 5.7% in K computer as compared to that of 20.1% by the conventional method.

Keywords: power prediction, HPC, deep learning, job scheduling

1. Introduction

Recent supercomputers consume much higher levels of power as their computing performance increases. The K computer, which was ranked the number one supercomputer on the TOP500 list in 2011, consumes 12.6 MW of power with performance of 10-PFlops. As of June 2020, the Fugaku computer, which was ranked the number one supercomputer on the TOP500 list, consumes 28.3 MW of power with performance of 415P-Flops [1], so the power consumption of the system is a critical limiter for the exascale computer systems [2]. Therefore, there are increasingly strong demands for lower-power computer hardware [3] and evolving energy management.

Power capping is an attractive method of energy management that controls total system power below a desired threshold. Previous research has put much effort into power capping by power-aware scheduling, which dynamically controls job scheduling using the predicted job power [4], [5]. Power for the queued jobs submitted into the scheduler is predicted utilizing available information, such as the job scripts. Most research leverages the fact that jobs with similar scripts will also have similarity in power. Although power capping with queued-job power prediction sometimes shows good performance, there are still big issues. One issue is that the accuracy of the queued-job power prediction decreases if there is no relationship between the job scripts and the power. Another issue is that the aggressive power capping decreases system utilization.

Recently, predictive control of cooling units has been studied to reduce total system energy in data centers [6], [7], [8]. Generally, cooling units are operated with enough cooling capacity for the maximum power of the system. This leads to overcooling of the cooling units in cases of low system utilization. Here, the time constant of the cooling-units stabilization after set points change is much longer than that of the changes in power. Predictive control can make each cooling unit operate efficiently in advance of the power fluctuations. Hence, a precise power prediction is necessary to achieve predictive power control. Unlike with conventional data centers, HPC power experiences MW-range changes, which makes highly precise power prediction difficult. A key challenge for predictive power control of HPC is the highly precise time-series prediction of total system power.

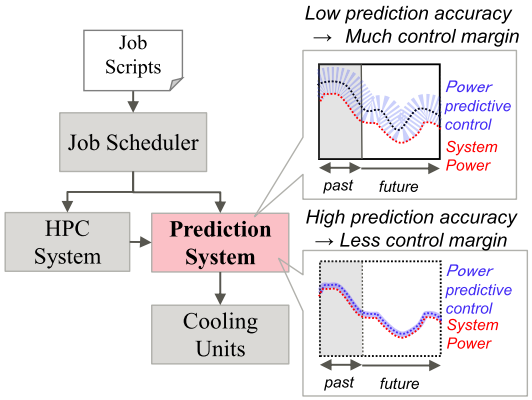


Fig. 1 Our proposed HPC system with predictive power control.

Figure 1 shows our conceptual HPC system based on predictive power control. User jobs are submitted and scheduled in a queue. Then, the queued jobs are executed in order and the system power increases. A prediction system predicts the future total system power. Cooling units are frequently controlled in advance with adjustments to the predicted system power variation. The

[1] Fujitsu Laboratories LTD., Kawasaki, Kanagawa 213–0012, Japan
[2] Fujitsu LTD., Kobe, Hyogo 650–0044, Japan
[3] RIKEN Center for Computational Science, Kobe, Hyogo 650–0047, Japan
[a)] shigeto.suzuki@fujitsu.com

power of the cooling units is almost proportional to the cooling capacity, so the power of the cooling unit can be reduced compared with the conventional static operation shown in Fig. 1.

We then briefly discuss the prediction period and the prediction accuracy for achieving efficient predictive power control. The targeted prediction period is dependent on time constants of the cooling units. The heat transfer processes in cooling units are prolonged (mostly within 30 minutes) compared to the IT load fluctuations [7], [8], [9]. Therefore, optimal cooling unit control considering heat transfer process can be realized by predicting system power up to 30 minutes in the future and controlling the cooling units based on it.

The targeted prediction accuracy relies on the power utilization of the HPC system. Typical HPC system utilization is below 100%. Moreover, job power per node is strongly dependent on each user's application, so the instantaneous power utilization of the systems is below 50% of the maximum system power capacity [10]. As the power prediction accuracy becomes worse, the power reduction with predictive control degrades due to ensuring a margin. The relative error of conventional power prediction based on job information is 21.1% (Section 3.2.1). The relative error of this work is 5.7% (Section 5). By reducing the cooling unit control margin, the cooling-units power can be reduced. For example, it applies predictive control to an environment with 3 MW of cooling units. The cooling unit requires about 1 MW of power to cool the power fluctuations of the computer. The relative error is 21.1% and 5.7%, control margins of 0.21 MW and 0.057 MW are required respectively. Therefore, it is possible to reduce 0.16 MW (5.3% of cooling-units power) by improving prediction accuracy.

In this paper, we propose a highly precise power prediction system, combining a queued-job prediction model and a run-job prediction model.

The rest of the paper is presented as follows: Section 2 provides analysis results about the ratio of queued jobs and run jobs in a real HPC system to reveal the importance of run-job prediction. Section 3 provides the data used in the experiment. Section 4 provides the proposed models. Section 4.1 shows our queued job prediction model based on a topic model, leveraging job entries extracted from the job scripts. Section 4.2 describes the run-job prediction model using job entries and power history of the job. We developed an integrated prediction model with two different characteristics of Machine Learning models. Section 5 reveals evaluation results of the combined power prediction system with queued and run jobs. We demonstrate total job power with a 5.7% prediction error by using 3 months of K computer operating logs. Related works are discussed in Section 6. Finally, we summarize this research in Section 7.

2. Analysis of Jobs

Generally, total system power is the sum of queued-job power and run-job power. **Figure 2** illustrates an example of job states at a certain time. At current time, 100% of the total system power consists of run jobs (Job2, Job3, Job6, Job7 and Job8). Job9 and Job10 are still queued jobs at current time, but will move to run states during the prediction period. At the current time, we need

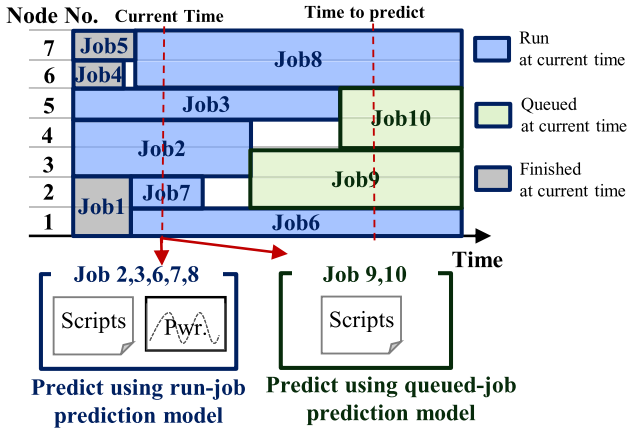


Fig. 2 An example of job states for power prediction.

to account for both the run jobs and the queued jobs for the power prediction.

Most of past research in HPC power prediction focused on the queued-job prediction because the purpose of the research was on power-aware scheduling or power capping (See Related Work section for details). Conventionally, the queued jobs are pre-scheduled based on the predicted power before they start running. As the run-job power ratio, however, is high, as we describe later in this section, it is important to predict them to achieve more accurate power-aware scheduling. Additionally, the cooling units can be controlled after the job has shifted to run states in the case of predictive power control. Therefore, the run-job power prediction is important for large-scale HPC sites.

This research describes an effective power prediction model of queued jobs and run jobs. The optimum power prediction model is different depending on the available input information, such as the job scripts and the power history. Job scripts that include user-specific textual information such as number of required nodes, user ID, job name and so on are available for the queued job. The run job can leverage the actual power history consumed by the job, in addition to the job script. We developed different types of models for queued and run jobs in order to achieve highly precise prediction as follows.

- Queued-job prediction model: predicting power of each job by utilizing job scripts when the job is in the queue state, will be described in Section 3.1.
- Run-job prediction model: predicting power of each job by utilizing job scripts and additional power history when the job is in the run state, will be described in Section 3.2.

Then, we statistically analyzed job states for power prediction by using K computer logs. **Figure 3** shows the power ratio of queued jobs and run jobs in the total job power from July to September in 2017. The horizontal axis indicates the future time to predict. The power-ratio evaluation was conducted every 30 minutes and average power ratio during each period was plotted on Fig. 3. When the time to predict is equal to 0, 100% of job power is run jobs. As the time to predict becomes longer, the power ratio of the queued jobs increases. Here, 30-minute future prediction is one of our targets for the predictive power control of the cooling units, as discussed in Section 1. From Fig. 3, 87% of jobs are still in run state in the 30-minute future. This indi-

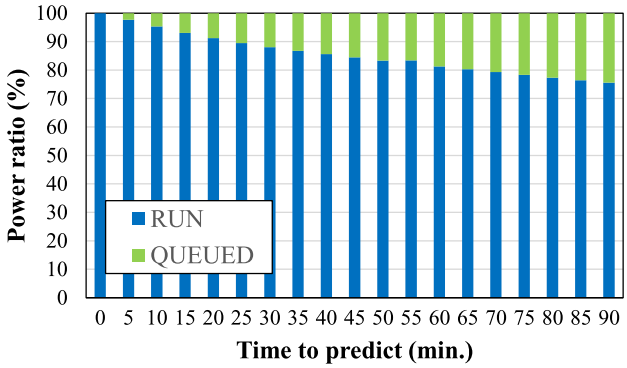


Fig. 3 Power ratio of queued jobs and run jobs depending on the prediction horizon.

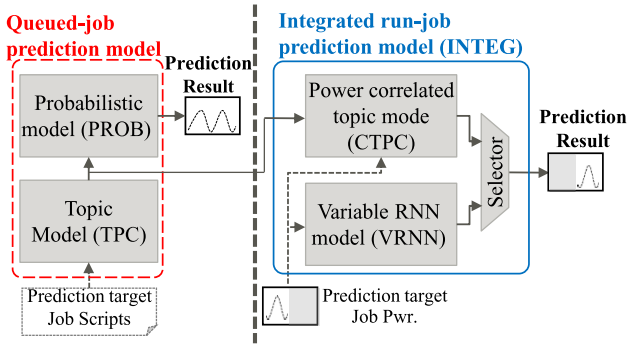


Fig. 4 A block diagram of our proposed power prediction system for HPC.

cates that run-job prediction is a significant part, especially for 30-minute future prediction.

In fact, the power ratio of queued and run is dependent on the average execution time of jobs. A longer average execution time means a larger percentage of run job in the total job. In our analysis using the K computer, the average execution time is 210 minutes. For other large-scale HPC sites, according to Refs. [11], [12], [13] average execution times were about 30 minutes, 110 minutes and 360 minutes respectively. This indicates that a relatively high run-job-power ratio can also be estimated.

3. Experiment Data

The experiment data is the operational information of the K computer. The K computer is a distributed memory supercomputer system consisting of 82,944 compute nodes and 5,184 I/O nodes [14]. In this paper, the power was obtained and calculated by using the monitored temperature of the CPU and inlet air flow, as shown in Ref. [15]. The CPU monitoring temperature and intake flow measurement interval is 5 minutes. Therefore, the power value calculated based on this is also a value every 5 minutes. And, each computer node's electric power fluctuation is predicted. The maximum power fluctuation for one node is 58W. In this paper, We call this the power per node. The power value of each node and the job information executed on that node were combined to obtain experimental data. Figure 6 shows the types of job information used in the experiment. The predict period is from July to September 2017, which has the largest power fluctuations of the year and is difficult to predict.

4. Power Prediction System

In this chapter, we describe the conventional power prediction methods using the topic model (TPC, PROB, CTPC), which use job information, and then discuss the proposed time-series power predicting technology (VRNN).

Figure 4 is a diagram of our proposed power prediction system. We developed a queued-job power prediction model (see Section 4.1), which includes a topic model (TPC) and a probabilistic model (PROB). The queued-job prediction model predicts the entire job power by using a job script as the prediction input.

We also developed a run-job power prediction model (see Section 4.2). The run-job prediction model predicts future job power by using the past job-power history and the TPC output as the prediction input. The run-job prediction model consists of two core prediction models, a power-correlated topic model (CTPC, see Section 4.2.1) and a variable RNN model (VRNN, see Section 4.2.2). The CTPC re-selects the prediction from 10 candidates of the TPC by comparing them with actual power history. The prediction, however will fail if there are no successful jobs in 10 candidates.

The VRNN can predict the power from a wide range of past job, but requires relatively long history to get an accurate prediction. During early execution time of the job, the CTPC, which mainly utilizes outputs of the TPC, shows higher precision. During late execution time of the job, the VRNN that was trained by a huge amount of power histories shows better performance. We developed an integrated run-job prediction model (INTEG, see Section 4.2.3), which dynamically selects one of the two models.

4.1 Queued-job Prediction Model

In this section, we describe the prediction method for queued jobs using the topic model [16]. The topic model is a widely used natural language processing method [17]. The information obtained from the queued job is only the submitted script. It is reported that jobs with similar job entries have similar power [18]. Power prediction techniques that choose the most similar job from past jobs by machine learning using the job entries in the submitted scripts have been proposed [19]. This approach requires manual weight tuning for each entry, in order to make more effective entries contribute more to prediction. Weights, however, could be different for each site.

We developed a two-step scheme using the topic model and the probabilistic model, which set proper weight for each entry automatically and enables high accuracy (**Fig. 5**). The topic model is trained from the past job entries extracted from the job scripts. In the prediction phase, it selects 10 candidates from past jobs based on their similarity to the target's job entries. The probabilistic model identifies the most similar job out of 10 candidates and uses its power as a prediction. Also, the execution time of most similar jobs is used as the prediction result of the execution time. The power for each node is predicted, and the power for each job is predicted by multiplying the predicted result by the number of nodes. This identification is based on the weights, which indicate probabilities of success. The weights are trained by the combination of the entries and power of past jobs. The weight training for each entry is determined by the following procedure. Job X and job Y are the past jobs identified by the topic model. Weights

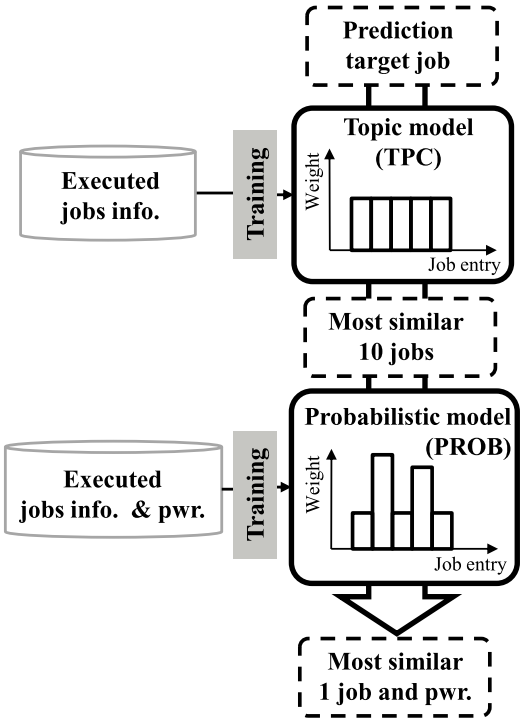
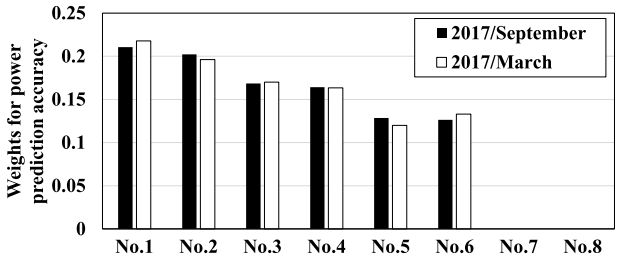


Fig. 5 A queued-job power prediction model with topic model and probabilistic model.



No.	job information	detail
No.1	request duration	Execution time requested by the job
No.2	number of request p node	Number of nodes requested by the job
No.3	user	User who submitted the job
No.4	group	Group to which the user belongs
No.5	job name	the job script name
No.6	queue	scale of the job (small, large, huge)
No.7	submission time	job submission time
No.8	job id	identification of the job

Fig. 6 Weight results on the K computer using the probabilistic model.

can be calculated from many X and Y pairs using the following algorithm.

i. Compare each entry between job X and job Y in training datasets and determine whether it matches or not.
ii. Compare time-series power between job X and job Y by using the Dynamic Time Warping (DTW) method, and score the probability of success based on their similarity.
iii. Based on the results of I and II, the probabilistic model calculates weight for each entry.

Figure 6 shows calculated weights of entries on the K computer. Six out of eight job entries used were relevant for power prediction accuracy.

Here, the probabilistic model succeeded in extracting more effective entries. For example, although both "Number of request node (No.2)" and "Queue (No.6)" represent the size of the job, No.2 should contribute more as it has more detailed information. Because of the probabilistic model, we confirmed that No.2 gets a higher weight than No.6.

10 candidates selected by the topic model are re-ranked, based on the probability of success calculated by the probabilistic model. Then, the first-ranked time-series power is selected as a prediction for the target job.

We evaluated the time-series power predictions of all jobs from July to September of 2017 on the K computer. Training was conducted using the past 3 months of data with a 15-minute interval. The reason for training every 15 minutes is that the queued-job prediction model predicts the power waveform by selecting a job that is similar to the past submitted script. It is desirable to update the model at as short an interval as possible, because jobs executed at a close time to the current job often have similar features to those of the current jobs [10]. We evaluated the interval dependency of the relative error. When the interval is changed to 15, 30, and 60 minutes, the relative error of September 2017 is 20.1%, 20.8%, and 22.4%.

Table 1 Precision comparison using relative error of total job power.

	July	August	September	**Total**
Only topic model (TPC)	17.8 %	22.0 %	23.6 %	**21.1 %**
Two-step model (TPC+PROB)	16.1 %	17.9 %	20.1 %	**18.0 %**

Table 1 shows the relative error of total job power prediction using only the topic model and using the two-step model. The two-step scheme is more accurate by 3.1% compared to the one-step, topic-model-only scheme. We achieved an average relative error of 18%. The configured weight adaptation contributes to an improvement in prediction accuracy.

4.2 Run-job Prediction Model

In this chapter, we propose the run-job prediction model. Section 4.2.1 describes the improved Topic model. We discuss VRNN in Section 4.2.2. We describe a model in which these are integrated.

4.2.1 Power-correlated Topic Model (CTPC)

We developed a run-job-power-prediction model called a power-correlated topic (CTPC) model, which utilizes the topic model in the queued-job prediction model (Fig. 4). The 10 candidates from the topic model are input into the CTPC instead of the probabilistic model, after the job is executed. The CTPC reselects the most matched one as the prediction result from the 10 candidates by comparing the actual power histories of the jobs and the predicted power. This reselection algorithm runs dynamically every 5 minutes (time step of the job power monitored). In this way, some of the jobs that failed in the queued-job prediction model change to successes.

The reselection algorithm of the CTPC is shown in Algorithm 1. Here, actual job power is acquired every 5 minutes. Time step i is the current time step from the start. $P(X)_0$ is the predicted power of the candidate X at the start time. $P(X)\_i$ is the predicted power of job X at time step i. X is a number from 1 to

Algorithm 1: CTPC

Run every 5 minutes (time step of the job power monitored)
Input: Predicted power of candidate X: $P(X)$, $P(X)_1$, ...$P(X)_i$
Actual power of the target: PA_0, PA_1, PA_i
Current time step is i
Output: Most similar candidate and future prediction power
for X=1 to 10 **do**
where i<6
then
return k=i
else
return k=6
calculate $SA_x = \left| \frac{1}{k} \sum_{j=1}^{k} (P(X)_{i-j} - PA_{i-j}) \right|$
return SA_Z
find the job X with minimum SA_X and
return future power of the job X as prediction result

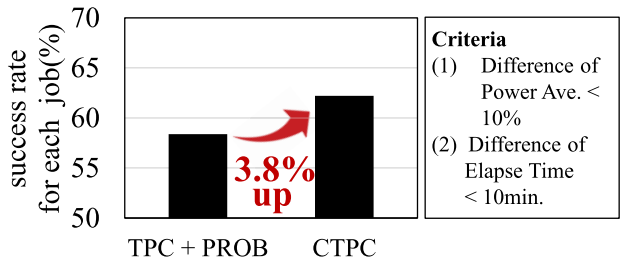


Fig. 7 Prediction comparison between the two-step queued-job prediction (TPC+PROB) and the power-correlated run-job prediction (CTPC).

10 which indicates the 10 candidates from the topic model. *PAi* shows actual monitored job power of the executed job at time step *i*.

We compare the power histories between the prediction candidates and the actual job, by going back from the current point. This maximum backwards step is set to 6 steps (meaning 30 minutes) for light calculation overhead. Simple averages (*SA*) of the difference between the actual power and the candidates were used for the evaluation function. This evaluation is executed dynamically every 5 minutes. When the actual job has a power history of over 6 steps, the *SA* of 6 points from current to 30 minutes before is calculated for all 10 candidates. When the actual job has a power history of less than 6 points, the *SA* of all time history is utilized for the evaluation. Then, the job that has the minimum *SA* is selected as the prediction result. This means that a candidate with the closest power history to the target job is selected.

We compared the queue-job prediction (TPC+PROB) in the previous section and the run-job prediction (CTPC) using the same datasets. Success rates of the prediction were compared for each job by using all jobs in the K computer from July to September in 2017. **Figure 7** shows the average success/failure rate. The success rate was evaluated using the criteria of both power and elapse time, as shown in Fig. 7. The CTPC model conducts the prediction dynamically every 5 minutes. We plotted the prediction of each job at a timing of 30 minutes from the start. The success rate is improved by 3.8% by adopting the power-correlated TPC compared with our best-case queued-job prediction.

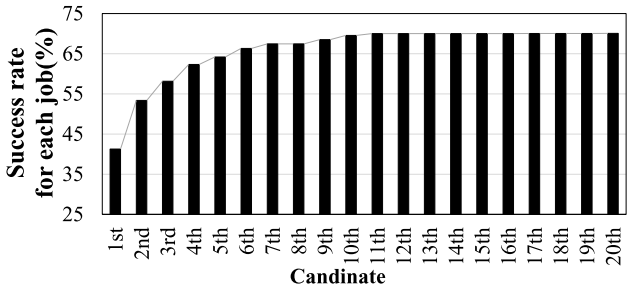


Fig. 8 The percentage of success for any job up to each candidate.

On the other hand, nearly 40% of jobs are still failed using the CTPC. We analyzed the cause of the failures. The CTPC selects one from only 10 candidates, and the prediction will fail if there are no successful jobs in these 10 candidates. **Figure 8** shows the percentage of success for any job up to each candidate. Up to 20th candidate, 30% of jobs are not successful.

E.g., the success rate of the 5th coordinate shows the percentage of jobs that have been successfully predicted for at least one job from 1st to 5th. We found out that there is no improvement in accuracy when increasing the number of the candidates. In order to further improve the prediction accuracy, a prediction model that can select the success power from a wide range of candidates is required.

4.2.2 Variable RNN Model (VRNN)

In this section, we propose another approach of the run-job-prediction model, leveraging a deep neural network. Today, deep learning technology achieves great success, especially in image classification fields [20]. Recurrent neural networks (RNN), which are able to learn the relationships between time series, have been studied for national language translation [21] and speech recognition [22]. Recently, the evolving RNN model is starting to be developed for forecasting of time series data [23], [24]. One of the great benefits of neural networks is that the networks can extract features from a huge dataset and cluster the datasets or find the similarities in the dataset. However, the main purpose of the research focuses on trend analysis of themselves using a long history of time-series data. On the other hand, the job power prediction for HPC is a different situation. The time series of jobs do not have yearly trends in themselves but they have similar trends in past jobs. The problem we should solve can be converted to the following question: "How do we identify the most similar pattern from the hundreds of thousands of patterns?"

Here, we proposed a run-job-prediction model leveraging the RNN model. First of all, we analyzed time-series data of jobs in the K computer because a certain number of similar patterns should exist in the past datasets for the RNN model. K-means clustering [25] was executed using the nearly 200,000 jobs from April to December in 2017. The maximum elapse time limit is 3 days and these time step of the jobs was 864, with 5-minutes each step. So, the K-means of 864 dimensions can cluster the job pattern based on the similarity of the job's power. **Figure 9** (a) shows the clustering accuracy depending on the number of clusters using the elbow method [26]. The vertical axis indicates the

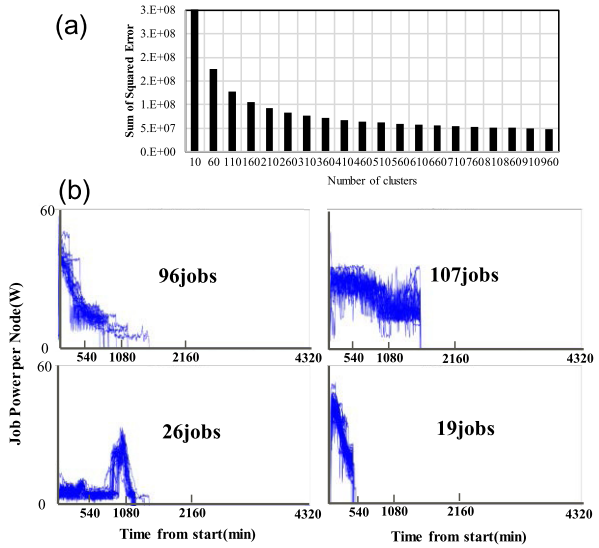


Fig. 9 (a) Elbow graph of K-means clustering depending on the clustering number. (b) Examples of power profiles in some clusters.

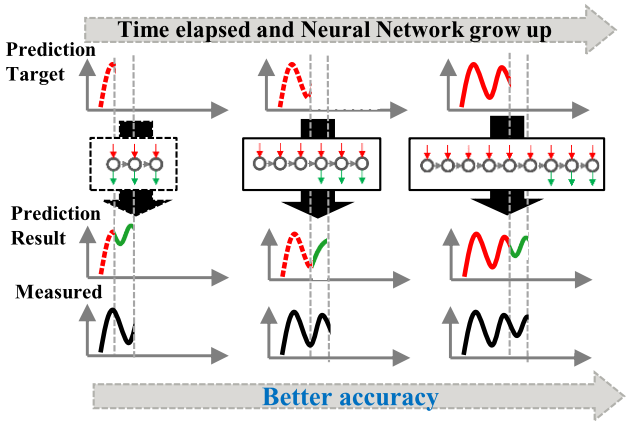


Fig. 10 Conceptual diagram of VRNN power prediction model.

sum of squared error for all time points. The all job power shape can be clustered into 400 to 800 clusters. Figure 9 (b) shows 4 examples of the clustered power shape in the case of 800 clusters. Each power profile can be successfully separated into 800 clusters. This means that an average of 250 similar power profiles exists in the datasets, which can be used for training of the neural network.

From Figs. 9 (a) and (b), we also observed a wide variety of power profiles in the datasets. This variation makes time-series prediction difficult. The neural networks need to identify the future job pattern from messy power profiles as shown in Fig. 9 (a). To solve this problem, we proposed a variable size RNN-based power prediction model (VRNN). **Figure 10** shows a conceptual diagram of the VRNN. The VRNN learn from the power profiles of past jobs. We designed it so that each neuron connected with the recurrent network accepts each timing of the job-power histories, that is, the first node accepts the power history of all job-start times. Each power history from the job start is input in parallel to the recurrent nodes to learn relationships between the time series from start to a certain time. The size of the network is extended to accept the entire power profiles of the job as the job executes, in order to extract the features from the power profiles. In the left picture in Fig. 10, few power histories are available for job prediction, so it is relatively difficult to identify the power profiles well. The network size grows depending on the available power histories in the right picture of Fig. 10, which results in better accuracy for power prediction. We will show the evaluation result in Fig. 10 later.

Algorithm 2: VRNN training algorithm

Run only once for training

Input：All job power for training

Power of job X at each time step: $P(X)_0$, $P(X)_1$, ... $P(X)_i$

Output：Trained model

```
/*Create Dataset*/
for n=1 to training job number do
    get maximum length of JOB X as iMax(X)
    if iMax(X) > 1
        x1= P(X)0
        y1 = P(X)1 ...to P(X)6
        store x1, y1 to DataSet1
    end if
    if iMax(X) > 2
        x2 = P(X)0, P(X)1
        y2 = P(X)2 ...to P(X)7
        store x2, y2 to DataSet2
    end if
    if iMax(X) > 4
        x4 = P(X)0, P(X)1, P(X)2, P(X)3
        y4 = P(X)4, ...to P(X)9
        store x4, y4 to DataSet4
    end if
    if iMax(X) > 6
        for Z=1 to int(iMax(X)/6) do
            x6Z = P(X)0 ...to P(X) 6Z-1
            y6Z = P(X)6Z ...to P(X) 6(Z+1)-1
            store x6Z, y6Z to DataSet6Z
        end for
    end if
end for
/*Training*/
for N=1 to max number of DatasetN do
    create sub-RNN ModelN and train using DatasetN
    store sub-RNN ModelN
end for
```

Algorithm 2 shows the training algorithm of the VRNN. First, datasets are created for the following training. Here, $P(X)_i$ is the power data of job X at time step i. In our algorithm, power histories of each job are reconstructed for training the many sub-models. For the sub-model of 5 minutes, $P(X)_0$ is used for training input and $\{P(X)_1, P(X)_2, P(X)_3, P(X)_4, P(X)_5, P(X)_6\}$ are used for training output as correct answer data. The jobs that already finish during the input period are eliminated from the training dataset because prediction of the finished jobs is not needed. The creation of the sub-models at 10 minutes and 20 minutes are executed in the same way. For the sub-model of 30 minutes, $\{P(X)_0, P(X)_1, P(X)_2, P(X)_3, P(X)_4, P(X)_5\}$ are used for training input and $\{P(X)_6, P(X)_7, P(X)_8, P(X)_9, P(X)_{10}, P(X)_{11}\}$ are used

Algorithm 3: VRNN inference algorithm

Run at i = 1, 2, 4, 6 and every 6 steps after that (12, 18, 24…)
Input: prediction target job power from start to current time step *i*
Output: predicted power from i+1 to i+7
search sub-RNN Modeli
    return sub-RNN Modeli
calculate prediction results using sub-RNN Modeli
    return prediction results from i+1 to i+7
if exist previous prediction power from i+1 to i+7
    overwrite that power
end if

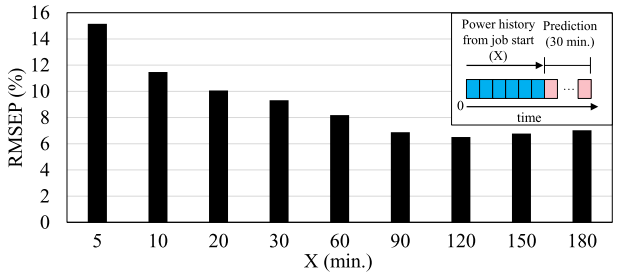


Fig. 11 Prediction comparison by the different sub-RNN model.

for training output as correct answer data. After that, the sub-models are created every 30 minutes. The size of the network in the sub-model gets larger depending on the size of the input data, which leads to better prediction accuracy.

Next, we explain the prediction procedure as shown in Algorithm 3. The prediction was executed for each job at 5, 10, 20 and 30 minutes, and every 30 minutes after that. We designed a sub-RNN model that can predict 30 minutes in the future (6 time steps) from the current power histories. Based on the available power history of the job, the optimally trained sub-model is loaded for the prediction. Each sub model was trained using the time histories of 5 minutes (meaning using just a single power history), 10 minutes, 20 minutes, 30 minutes, and every 30 minutes thereafter (60, 90 minutes, etc.). In the case of 10, 20 and 30 minutes, there are previous prediction results. The previous results are overwritten by the new results because the model trained from a longer job pattern shows better accuracy.

In order to confirm the usefulness of the VRNN model, we did an experiment using the prediction testbed shown in **Fig. 11**. 24,991 jobs in April 2017 in the K computer were used for training, and all jobs from July to September 2017 were predicted. The power for each node is predicted, and the power for each job is predicted by multiplying the predicted result by the number of nodes. Also, only one node's power is used for one job in training. Prediction period was fixed to 30 minutes and input power histories were changed from 5 to 180 minutes. Depending on the input power history, different sub-RNN models are utilized for the prediction. The root mean square errors percentage (RMSEP), which is normalized RMSE, is shown using the following equation.

$$RMSEP = \frac{1}{P_{max}} \frac{1}{m} \sum_{x=1}^{m} \sqrt{\frac{1}{n} \sum_{i=1}^{n} (A(x)_i - P(x)_i)}$$

"$A(x)_i$" is the actual power of job "x" at time step "i". "$P(x)_I$" is the predicted power of job "x" at time step "i". "P_{max}" is 58, which is maximum power by watt per node. "m" is number of total jobs. The RMSEP indicates normalized RMSE of prediction and actual power, averaged by all the jobs. The horizontal axis is power history from the start of the job. The RMSEP improves drastically from 0 to 90 minutes. This result indicates that the longer sub-RNN model with the longer input data shows better accuracy. The RMSEP becomes almost stable from 90 to 180 minutes. This implies that over 90 minutes of power history from the start is enough to obtain good prediction accuracy. We confirmed the benefit of the VRNN, which changes the model size depending on the input power.

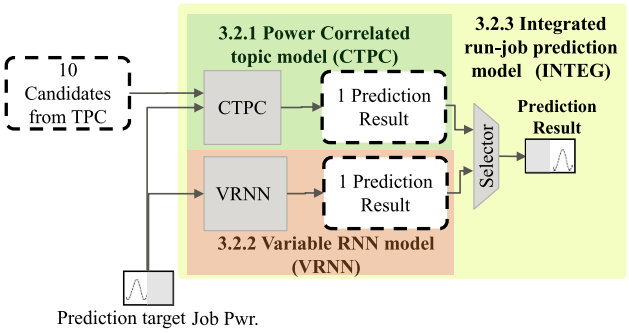


Fig. 12 An integrated run-job prediction model consists of the CTPC and the VRNN.

4.2.3 Integrated Run-job Prediction Model

In this section, we propose an integrated run-job prediction model (INTEG) that includes the CTPC shown in Section 4.2.1 and the VRNN shown in Section 4.2.2. Here, each model has different features. The VRNN model can predict the job power accurately by learning the huge power patterns. Nevertheless, the weak point of the VRNN model is prediction in short available job time steps. On the other hand, the CTPC basically uses a topic model output that generates a relatively good prediction result using the job entries, even in the case of a short power history. Therefore, we developed the INTEG, which can support each model's weak points.

Figure 12 shows the conceptual diagram of the run-job prediction model, the CTPC, the VRNN and the INTEG. The prediction results of the VRNN that learned from a huge amount of power data from past jobs are compared with the prediction results of the CTPC. The INTEG choices the better prediction results by using a similar algorithm to Algorithm 1. Each job can dynamically leverage the best prediction model every 5 minutes.

We compared the prediction of each run-job model, i.e., the CTPC and the VRNN model. **Figure 13** shows prediction results of 6 different-shape jobs for comparison. The left side of the figure (job A, B, and C) are examples of a job in which the prediction results of the CTPC are better in terms of prediction accuracy than that of the VRNN. The CTPC accurately predicts the power change points if there is a power waveform similar to the prediction target in 10 candidates. The right side of the figure (Job D, E, F) are examples of a job in which the prediction results of the VRNN are better in prediction accuracy than that of the CTPC. When the CTPC fails in the case of no success data in 10 candidates, the VRNN model predicts highly accurate results.

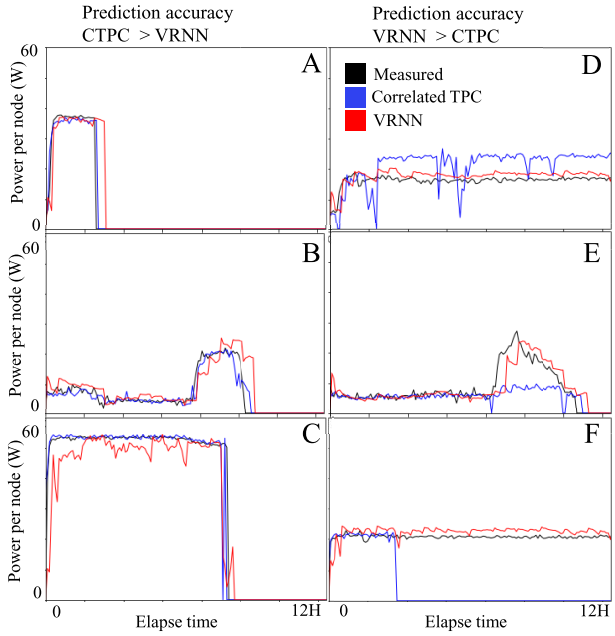


Fig. 13 Examples of prediction for 6 different job. Actual data, the VRNN model and the CTPC are shown in comparison.

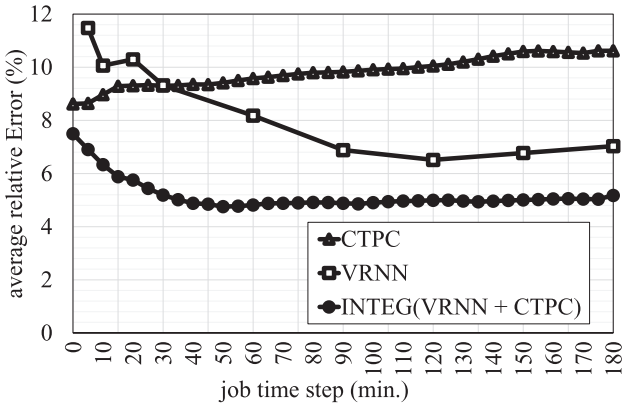


Fig. 14 Prediction error comparison of the CTPC model, the VRNN model and the integrated model.

The VRNN does not fail completely because of the wide learning from the past huge datasets.

Figure 14 shows a comparison of prediction accuracy depending on the available job time steps. All jobs executed from July to September 2017 on the K computer were predicted by each model. The training interval of each model is in **Table 3**. The CTPC, the VRNN and the INTEG were compared with the average relative prediction error. The relative errors in each time step for all jobs were calculated. The horizontal axis shows job time step from the job's start time for the predicted jobs. A large value of the time step means large available power histories for the prediction. The vertical axis shows the average relative errors at the prediction timing for all jobs. The error was evaluated up to 30 minutes in the future at each time step, and then averaged for all the jobs.

Interestingly, the prediction-time-step dependencies of each model are quite different. The relative prediction error of the VRNN decreases as the prediction time step increases. This is because the VRNN can extract features from the long power steps, as described in previous section. On the other hand, the relative prediction error of the CTPC very slightly increases as the prediction time step increases. The prediction-time-step dependency is lower than that of the VRNN model. Before 15 minutes, the prediction error of the CTPC changes rapidly. This is because the reselection of the prediction is based on the few actual time points in this period as shown in Algorithm 1. Nevertheless, the relative error is lower than that of the VRNN. After 30 minutes, the reselection of the CTPC was always done by using the previous 30-minutes power data. The prediction error of the CTPC gradually increases at a constant rate during this horizon. And in this horizon, the VRNN model is significantly lower than the CTPC. Here, this gradual increase is also observed for the VRNN in the period after 120 minutes. We hypothesize that this is because there are few training datasets (only long jobs) in this period, and the model needs to extract an answer from a small amount of training data. Detailed analysis will be done in our future work.

The INTEG achieves the selection of the optimum model for each job on each job time step. In Fig. 14, the result of the INTEG is lower than the simple combination of both results (the CTPC and the VRNN). The prediction error from the integration model shows less than 6% relative error after 15 minutes and becomes very stable after 30 minutes. Also, even at the start time of jobs, the INTEG shows the lowest relative error at below 8%. In the INTEG, the selection of the CTPC and the VRNN is dynamically conducted at each prediction timing every 5 minutes. The relative error curve of the INTEG is very smooth in Fig. 14. Determining which model to adapt dynamically may cause the model to change suddenly, so the relative error will not be smooth. By adapting Algorithm 1 so that the selection is judged using the average value of the past 30-minutes power time steps, the INTEG relative error becomes very smooth.

Here, we summarize the run-job prediction model. Soon after the job start, the CTPC is frequently selected by the INTEG as the prediction model. The VRNN, which cannot identify the power shape from a small number of power time steps, shows inaccurate predictions during this period. After 30 minutes from the start, the VRNN model is frequently selected as the prediction model instead of the CTPC. However, the CTPC is infrequently selected in this period. This is because the VRNN is a kind of regressive model that sometimes generates a slight error. We estimate that this is caused by the noise in the training datasets. On the other hand, the CTPC is a reselection algorithm of the 10 candidates, which is basically the past power shapes. The prediction results sometimes completely coincide with the actual data because the same user sometimes executes exactly identical jobs in the HPC field.

5. Evaluation

This chapter shows the evaluation of our developed prediction model using the K computer's total job power. **Table 2** shows the computer environment used for training and prediction. We used the Nvidia GPU P100 for the training of the VRNN only. The training of the queued-job prediction (TPC+PROB) and the CTPC were done by the CPU.

Table 3 shows the training conditions, which consist of training data, training interval, and training time. We demonstrated

Table 2 Machine specification used for this prediction.

Compute Server	
CPU	Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz 2sockets, 14core/socket
Memory	2400 RDIMM 128GB
System disk	SAS 12Gbps, 10krpm
GPU	NVIDIA P100 2slot

Table 3 Training condition.

	Queued-job prediction model (TPC or TPC+ PROB)	Run-job prediction model	
		CTPC	VRNN
Training Data	Past 3 months all jobs (60,000~80,000 jobs)	None (Utilize the queued-job prediction result)	2017/April all jobs (24,991 jobs)
Training interval	15min.	None	Only once
Training time	5min.	None	78 hours

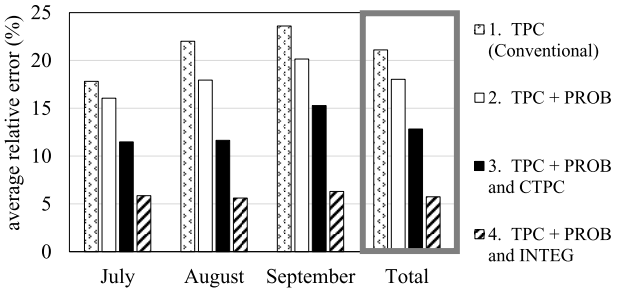


Fig. 15 Prediction results of total job power. Four predication model results are shown in comparison.

the prediction from July to September 2017. The power of this 3 month has the biggest standard deviation in 2017 and is the most difficult to predict. The queued-job prediction model used all jobs in the past 3 months for the training. The queued-job prediction model was updated every 15 minutes using the past 3 months from the current time, including newly submitted jobs set to the training interval shifted every 15 minutes.

The CTPC uses the prediction result of the queued-job prediction model, so there is no need for training. The VRNN model trains only once and does not update the model in this evaluation. The learning time of the VRNN was 78 hours using the GPU. The 1-month training interval (April) is separate from the prediction period (July to September). We assume the prediction accuracy will be higher if the training time is closer to the prediction time. However, the VRNN precision relative error is as small as 5.7%.

Figure 15 shows the evaluation results of predicted total job power in the K computer. The evaluation period is July to September 2017. The number of jobs predicted was 81,500. The power of all jobs executed in the evaluation period was predicted, and the total job power was calculated depending on their executed time in real logs from the K computer. The total power

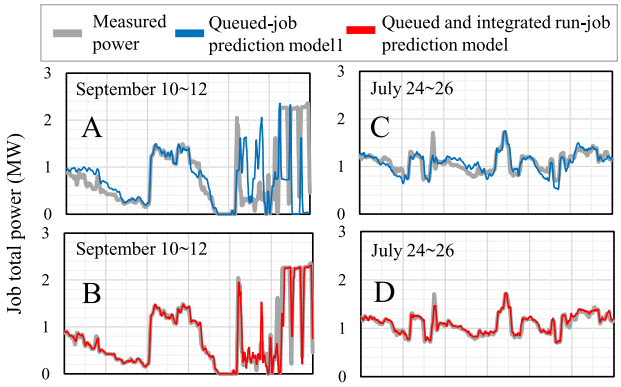


Fig. 16 Examples of total job power predict. A, C: TPC prediction (September 10 to 12 and July 24 to 26, respectively), B, D: TPC + PROB + INTEG prediction (Same period with A, C).

of the measured job power was also calculated in the same way. Then, the total predicted job power and the total measured job power were compared. The relative error was used for the evaluation.

The vertical axis in Fig. 15 shows the monthly average of the obtained relative errors and total average relative errors. We compared four model cases:

1. TPC (Conventional)
The TPC model predicted both queued jobs and run jobs in the case of the queued-job prediction model only (Section 3.1).

2. TPC + PROB
The TPC + PROB model predicted both queued jobs and run jobs in the case of the queued-job prediction model only (Section 3.1).

3. TPC + PROB and CTPC
The queued-job prediction was conducted by the TPC + PROB. Model and the run-job prediction was executed by the CTPC (Section 3.2.1).

4. TPC + PROB and INTEG
The run-job prediction model in 3 was changed from the CTPC to the INTEG (Section 3.2.3).

The prediction was run every 30 minutes and the decision for job attribution (which job is queued and which is run) was also calculated every 30 minutes. Then, all calculated data was averaged by the prediction period.

In Fig. 15, all proposed algorithms have better prediction accuracy than TPC. Comparing the TPC with the INTEG, the prediction accuracy improved by 14.4%. The same trend of improvement was observed in the monthly relative errors. The improvement was independent on the prediction horizon. Finally, an average relative error of 5.7% was achieved. We believe that the prediction error is dependent on the sites, the length of the prediction period and the evaluation function for accuracy. These cannot be compared directly, even though our result is the most accurate job power prediction system, demonstrated over long prediction period using large-scale real HPC data. Details are described in related work.

Figure 16 shows the examples of the job total measured power and the predicted power. The power value is not an instantaneous value but an average value for 30 minutes. This is because the

average value of power for 30 minutes, which is the time constant, is the control index for cooling-units control. In the period where the power change is small (July 24 to 26), both the TPC and the INTEG predict with high accuracy (C, D). Improvement was observed in the case of the INTEG. If the prediction of the TPC fails, the prediction continues to fail until job completion, and the error is large. It can be well observed that the prediction succeeds by predicting and correcting this with the INTEG (A, B). The prediction that failed in the TPC is changed to a success by being corrected in the INTEG.

Overhead for job-scheduler by the prediction is short, i.e., less than 0.1 sec per 100 jobs.

6. Related Work

Several methods have been proposed to predict job power of predominantly queued jobs or resource usage for power-aware job scheduling. A method of predicting job power using machine learning was reported [4]. A random forest model trained by 100,000 jobs selects a job script similar to the prediction target and uses the power of the selected job as the predicted power, by using the Aurora-supercomputer logs. The average prediction error for each job was calculated to be 5%. The job power consumption, not time-series power, was used for the evaluation. The size of the test set was limited to 943 and 714 jobs. We demonstrate the queued-job prediction of 3-month all jobs (81,500 jobs) by using the topic model. A power-prediction for power capping by statistic model was reported using the Luna super computer [27]. The parameters of the statistic model are determined using a hierarchical Bayesian model. The power required for the job is predicted with a statistical model in which the determined parameters are substituted. A power prediction using a dynamic learner for a power-aware scheduler was reported [10]. They utilized the IBM Blue Gene/Q workload trace. The dynamic learner calculates power using a simple rule, in which the previous power of the same user's job is used as the prediction power. This kind of trend differs from one HPC site to another. Our queued-job prediction model enables auto-weight-tuning of the job entries by the probabilistic model. An application-prediction technique was reported because jobs with same applications had similar power profiles [28]. The paragraph vector model clustered jobs from the hashes and symbols in the scripts to 328 kinds of applications. The demonstration was carried out on the K computer between September 2016 and March 2017 (273,121 jobs). The application predicted with an accuracy of approximately 92%. This work was only focused in the application prediction, not power prediction like this work. All above works were conducted for queued-job power prediction.

In terms of time series prediction, run-job power prediction has hardly been conducted for HPC. Early work has shown that RNNs outperform feedforward networks and various types of linear statistical models on general time series [29]. Subsequently, various RNN-based models were developed for different time series, as noisy foreign exchange rate prediction [30], chaotic time series prediction in communication engineering [31] or stock price prediction [32]. Also recently, a Long-Short Term Memory (LSTM), which is an RNN, with an evolving likelihood model was reported for general-purpose time-series forecasting [24].

The difference between our work and these existing RNN models is that the conventional RNN have been typically applied to individual time series; i.e., a different model is fit to each time series independently [33]. The RNN predicts the future from the trends that are learned from relatively long past histories of themselves (monthly, yearly for weather forecasts and stock markets). On the other hand, this work can be trained using a huge number of relatively short job power profiles (minutes, hours). This is quite a different development compared to previous works. Instead of learning the trend, our VRNN learns similarity from similar past profiles.

A multi-branch LSTM for stock price forecasting was reported [34]. They utilized different LSTM models for different patterns to learn each part of the stock market data. Each LSTM was taught from 4 to 5 sets of clustered data by K-means. This approach is not suitable in our case, as we have 400 to 800 kinds of shapes of job profiles, which cannot be identified by K-means until the job finishes. Our integrated run-job prediction model can identify the future power without the pre-clustering.

7. Conclusion and Future Work

We developed a highly accurate job total power predicting system to predictively control the cooling units of a large scale HPC system. High precision prediction with an average relative error of 5.7% was demonstrated for all jobs over 3 months on the K computer. We developed a queued-job power prediction model that consists of the topic model and the probabilistic model. A benefit of the queued-job prediction model is easy introduction without parameter tuning. We also proposed a noble integrated run-job prediction model that combines the power-correlated topic model and the variable RNN model. These two run-job models were successfully integrated in order to achieve high-precision prediction, shoring up each other's weak points in job execution. Furthermore, to the best of our knowledge this work is the first large-scale attempt at combining queued-job power prediction and run-job power prediction to predict the total power of the HPC system.

As future work, we will adopt the proposed prediction model to other HPC systems to confirm its generality. Especially, the impact of process fluctuations needs to be discussed carefully. In this result, since many jobs were executed on many nodes, the power per job was averaged, so it did not affect the prediction accuracy. However, in an HPC system with a majority of jobs executed on small nodes. The accuracy of system power prediction can be degraded.

We also plan power control experiments of the cooling unit and job scheduler by using it.

References

[1] TOP500 Lists, available from ⟨https://www.top500.org/lists/2020/06/⟩.
[2] A The race to exascale: A story of superpowers and supercomputers, available from ⟨https://www.datacenterdynamics.com/analysis/superpowers-supercomputers-and-race-exascale/⟩.
[3] Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al.: In-datacenter performance analysis of a tensor processing unit, *Proc. 44th Annual International Symposium on Computer Architecture*, pp.1–12, ACM

(2017).
[4] Borghesi, A., Bartolini, A., Lombardi, M., Milano, M. and Benini, L.: Predictive Modeling for Job Power Consumption in HPC Systems, *High Performance Computing: 31st International Conference, ISC High Performance 2016* (2016).
[5] Whitepaper: Energy and Power Aware Job Scheduling and Power Management, Energy Efficient HPC Working Group, Working draft, available from ⟨https://eehpcwg.llnl.gov/documents/conference/sc17/sc17_bof_epa_jsrm_whitepaper_110917⟩.
[6] Endo, H., Kodama, H., Fukuda, H., Sugimoto, T., Horie, T. and Kondo, M.: Effect of climatic conditions on energy consumption in direct fresh-air container data centers, *IEEE 4th International Green Computing Conference* (*IGCC*) (2013).
[7] Tarutani, Y., Hashimoto, K., Hasegawa, G., Nakamura, Y., Tamura, T., Matsuda, K. and Matsuoka, M.: Temperature distribution prediction in data centers for decreasing power consumption by machine learning (Dec. 2015).
[8] Serale, G., Fiorentini, M., Capozzoli, A., Bernardini, D. and Bemporad, A.: Model predictive control (MPC) for enhancing building and HVAC system energy efficiency: Problem formulation, *Applications and Opportunities Energies*, Vol.11, p.631 (2018).
[9] Chai, Y., Wu, A., Dong, N., Wang, Y. and Li, Y.: Dynamic Operation and Control Strategy of Abcorption Chiller under different working Conditions, *Proc. 13th World Congress on Intelligent control and Automation* (2018).
[10] Wallace, S., Yang, X., Vishwanath, V., Allcock, W.E., Coghlan, S., Papka, M.E. and Lan, Z.: A data driven scheduling approach for power management on hpc systems, *SC16: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp.656–666 (Nov. 2016).
[11] Zasadziński, M., Muntés-Mulero, V., Solé, M. and Ludwig, T.: Mistral Supercomputer Job History Analysis (2018), available from ⟨https://arxiv.org/abs/1801.07624⟩.
[12] Meneses, E., Ni, X., Jones, T. and Maxwell, D.: Analyzing the Interplay of failures and Workload on a Leadership-Class Supercomputer, *Cray User Group Conference* (2015).
[13] Joubert, W. and Su, S.: Application workloads on the jaguar cray XT5 system, *Cray User Group Conference* (2012).
[14] Yamamoto, K., Uno, A., Murai, H., Tsukamoto, T., Shoji, F., Matsui, S., Sekizawa, R., Sueyasu, F., Uchiyama, H., Okamoto, M., Ohgushi, N., Takashina, K., Wakabayashi, D., Taguchi, Y. and Yokokawa, M.: The K computer Operations: Experiences and Statistics, *Proc. International Conference on Computational Science* (*ICCS*) (2014).
[15] Uno, A., Hida, H., Inoue, F., Ikeda, N., Tsukamoto, T., Sueyasu, F., Matsushita, S. and Shoji, F.: Operation of the K computer Focusing on System Power Consumption, *IPSJ Trans. Advanced Computing Systems*, Vol.8, No.4, pp.13–25 (Nov. 2015).
[16] Suzuki, S., Hiraoka, M., Shiraishi, T., Fukuda, H., Yamamoto, T., Matsui, S. and Uno, A.: Power prediction with probabilistic topic modeling for HPC, *ISC2019 HPC RESEARCH POSTER* (2019).
[17] Papadimitriou, C., Raghavan, P., Tamaki, H. and Vempala, S.: Latent Semantic Indexing: A probabilistic analysis (Postscript), *Proc. ACM PODS*, pp.159–168, DOI: 10.1145/275487.275505, ISBN 978-0897919968 (1998).
[18] Netti, A., Galleguillos, C., Kiziltan, Z., Sirbu, A. and Babaoglu, O.: Heterogeneity-aware resource allocation in HPC systems, *Proc. ISC'18*, Vol.10876 of Lecture Notes in Computer Science, pp.3–21, Springer (2018).
[19] Le, Q., Ranzato, M., Monga, R., Devin, M., Corrado, G., Chen, K., Dean, J. and Ng, A.: Building high-level features using large scale unsupervised learning, *Proc. ICML*, pp.81–88 (2012), available from ⟨http://research.google.com/archive/unsupervised_icml2012.pdf⟩.
[20] Sutskever, I., Vinyals, O. and Le, Q.V.: Sequence to sequence learning with neural networks, *Advances in Neural Information Processing Systems*, pp.3104–3112 (2014).
[21] Alex, G., Abdel-Rahman, M. and Geoffrey, H.: Speech recognition with deep recurrent neural networks, *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (*ICASSP*), pp.6645–6649, IEEE (2013).
[22] Zhang, G., Patuwo, B.E. and Hu, M.Y.: Forecasting with artificial neural networks: The state of the art, *International Journal of Forecasting*, Vol.14, No.1, pp.35–62 (1998).
[23] Flunkert, V., Salinas, D. and Gasthaus, J.: DeepAR: Probabilistic forecasting with autoregressive recurrent networks, arXiv preprint arXiv:1704.04110 (2017).
[24] Steinhaus, H.: Sur la division des corps materiels en parties (French), *Bull. Acad. Polon. Sci.*, Vol.4, No.12, pp.801–804, MR 0090073, Zbl 0079.16403 (1957).
[25] Yamamoto, K., Tsujita, Y., Uno, A.: Classifying Jobs and Predicting Applications in HPC Systems, *ISC High Performance 2018: High Performance Computing*, pp.81–99 (2018).
[26] Thorndike, R.L.: Who Belongs in the Family?, Psychometrika, Vol.18, No.4, pp.267–276, DOI: 10.1007/BF02289263 (Dec. 1953).
[27] Storlie, C., Sexton, J., Pakin, S., et al.: Modeling and predicting power consumption of high performance computing jobs, arXiv preprint arXiv:1412.5247 (2014).
[28] Li, Y., Hu, H., Wen, Y. and Zhang, J.: Learning-based power prediction for data centre operations via deep neural networks, *Proc. 5th International Workshop on Energy Efficient Data Centres*, No.6, ACM (2016).
[29] Connor, J.T., Atlas, L.E. and Martin, R.D.: Recurrent networks and NARMA modeling, Advances in Neural Information Processing Systems 4, Moody, J.E., Hanson, S.J. and Lippmann, R.P. (Eds.) San Matteo, CA: Morgan Kauffman, pp.301–308 (1992).
[30] Giles, C.L., Lawrence, S. and Tsoi, A.C.: Noisy time series prediction using recurrent neural networks and grammatical inference, *Machine Learning*, Vol.44, No.1-2, pp.161–183 (2001).
[31] Jaeger, H. and Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, *Science*, Vol.304, No.5667, pp.78–80 (2004).
[32] Hsieh, T.J., Hsiao, H.F. and Yeh, W.C.: Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm, *Applied soft Computing*, Vol.11, No.2, pp.2510–2525 (2011).
[33] Langkvist, M., Karlsson, L. and Loutfi, A.: A review of unsupervised feature learning and deep learning for time-series modeling, *Pattern Recognition Letters*, Vol.42, pp.11–24 (2014).
[34] Shao, X., Ma, D., Liu, Y. and Yin, Q.: Short-term forecast of stock price of multi-branch LSTM based on K-means, *4th International Conference on Systems and Informatics* (*ICSAI*), pp.1546–1551 (2017).

Shigeto Suzuki is a researcher at Fujitsu Laboratories Ltd., Japan. His responsibilities at Fujitsu are the machine-learning design for ICT operation and system development for AI workload. He received his B.E. degree from Shibaura Institute of Technology in Electronic Engineering, in 2009, and his M.S. Degrees in Electronics and Applied Physics from Tokyo Institute of Technology in 2011.

Michiko Hiraoka is a researcher at Fujitsu Ltd., Japan. Her responsibilities at Fujitsu are the operation management of high-performance computing systems. She received his B.E. degree from Kobe University, in 2011, and his M.S. Degrees in Agriculture from Kyoto University in 2013.

Takashi Shiraishi is a senior researcher at Fujitsu Laboratories Ltd., Japan. His responsibilities at Fujitsu are the machine-learning design for ICT operation and system development for AI workload. He received his B.E. Degree in Physics, and his M.S. Degrees in Science and Technology from Tsukuba University in 1998 and 2002 respectively.

Enxhi Kreshpa is currently a researcher in Fujitsu Laboratories Ltd., Japan. She has interests in artificial intelligence applications in supercomputers, and also, in computer networks. She received her M.S. Degrees from the Polytechnic University of Tirana, Albania in 2017 and after that joined Fujitsu Laboratories Ltd. in 2018.

Takuji Yamamoto received his B.S. degree from Keio University in 1986. In 1986, he joined Fujitsu Laboratories Ltd. Japan, where he is currently a Senior Expert. He has been engaged in research on high-speed communication systems and computing systems. His current interests include the next generation computing with high performance and low power.

Hiroyuki Fukuda received his B.S. degree in Chemistry from Kanazawa University, Kanazawa, Japan in 1987. He joined Fujitsu Laboratories Ltd., Atsugi, Japan in 1987, where he has been engaged in facilities management systems. He is a member of the Society of Heating, Air-Conditioning and Sanitary Engineers of Japan.

Shuji Matsui is a Deputy Head of Computational Science Div. at Fujitsu Ltd Technical Computing Business Unit. His responsibilities at Fujitsu are the operation management of high-performance computing systems.

Masahide Fujisaki is executive architect at Fujitsu Limited and Digital Annealer project of Fujitsu Laboratories Ltd. He joined Fujitsu Limited in 1984. His primary research interests are optimization of high-performance simulation and solvers on various type of HPC platform.

Atsuya Uno is a unit leader of System Operations and Development Unit in Operations and Computer Technologies Division at RIKEN Center for Computational Science.