A GRU-Based Prediction Framework for Intelligent Resource Management at Cloud Data Centres in the Age of 5G

Yao Lu, *Member, IEEE,* Lu Liu, *Member, IEEE,* John Panneerselvam, Bo Yuan, *Member, IEEE,* Jiayan Gu, *Member, IEEE,* Nick Antonopoulos

***Abstract*—The increasing deployments of 5G mobile communication system is expected to bring more processing power and storage supplements to Internet of Things (IoT) and mobile devices. It is foreseeable the billions of devices will be connected and it is extremely likely that these devices receive compute supplements from Clouds and upload data to the back-end datacentres for execution. Increasing number of workloads at the Cloud datacentres demand better and efficient strategies of resource management in such a way to boost the socio-economic benefits of the service providers. To this end, this paper proposes an intelligent prediction framework named IGRU-SD (Improved Gated Recurrent Unit with Stragglers Detection) based on state-of-art data analytics and Artificial Intelligence (AI) techniques, aimed at predicting the anticipated level of resource requests over a period of time into the future. Our proposed prediction framework exploits an improved GRU neural network integrated with a resource straggler detection module to classify tasks based on their resource intensity, and further predicts the expected level of resource requests. Performance evaluations conducted on real-world Cloud trace logs demonstrate that the proposed IGRU-SD prediction framework outperforms the existing predicting models based on ARIMA, RNN and LSTM in terms of the achieved prediction accuracy.**

***Index Terms*—5G, Internet of Things, Resource Management, Cloud Computing**

I. Introduction

INTERENTof Things(IoT) is undergoing rapid development in the recent years, and the emergence of 4G networks has led to the invention of Internet of Everything (IoE). The increasing number of mobile devices being connected to the 4G network naturally causes technical bottleneck issues, which in fact limits the efficiencies of 4G. Given the recent emergence of 5G networks, the way of using mobile phones will undergo a blowout transformation. Recent mobile systems are characterising reasonable compute provisioning, high speed, ultra-reliability and low-latency [1]. Major countries in the worlds are already in the process of deploying, promoting and commercialising 5G networks. IoT applications will face a surge with the deployments of 5G network, benefitting smart home, smart vehicles, smart cities, and thousands of handheld devices [2], as shown in Fig. 1 According to the Gartner, up to 20.4 billion IoT devices will be connected through machine-to-machine communication networks [3].

But the increasing trend in the number of IoT devices is always a source of causing network issues such as bottleneck, congestion, and contention etc. Cisco reports that the mobile data traffic will increase by 53% annually up until 2020 [4]. Such an overwhelming mobile traffic data requires efficient strategies of data management. The paradigms of edge computing and fog computing provides supplements for efficient scheduling and management of data, in particular facilitates local processing of the IoT data, through exploiting resources in the edge networks. Although edge devices facilitates such local processing of data, the involvement of back-end datacentres cannot be eliminated, due to the intense processing requirements of heavy-weight IoT applications. Moreover, resource constraint edge devices may not provide adequate computational and caching services due to their limited hardware resources. The devices in fog layers usually comprise the storage and computing resources deployed in a macro base station [5]. In this sense, the resources in the fog layer usually characterise resources better than those of stand along mobile devices, but fewer than the back-end datacentre resources in the Cloud. Although fog computing infrastructure can provide considerable caching and computational services, it still suffers from limited storage and computing resources whilst processing complex tasks [6]. Thus, edge and fog computing cannot be regarded as a substitute for Cloud computing, but can be used in conjunction with Cloud in order to enhance the functional efficiency of the entire network, as shown in Fig. 1. Resource management in the Cloud is an ongoing problem, with the extension of fog and edge computing, this issue is increasing complexity

One of the prevailing issues of Cloud Computing is the energy consuming nature of its resources such as massive servers, cooling supplements, lighting etc., all require enormous input energy and also datacentres are regarded as a major source environmental pollution [7]. Cloud service providers are expected to provide high-quality services and to meet negotiated SLA (Service Level Agreement) with the clients. One of the primary reasons for the increasing operational costs of Cloud providers is their strategy of over-provisioning the resource requirements of the workloads. Such over-provisioning level of resources such as CPU, memory, bandwidth etc., usually far exceed the actual requirements of the workloads.

• Yao Lu and Lu Liu are with the School of Informatics, University of Leicester, UK. Yao Lu is also with School of Computer Science and Telecommunication Engineering, Jiangsu University, Jiangsu, China.

• John Panneerselvam, Bo Yuan and Jiayan Gu are with the School of Electronics, Computing, and Mathematics, University of Derby, UK.

• Nick Antonopoulos is with University Executive Office, Edinburgh Napier University, EH11 4DY, UK

• Lu Liu is the corresponding author. Email: l.liu@leicester.ac.uk

On the other hand, reducing the level of resource provision might risk workload failures and termination under certain scenarios such as resource scarcity, overwhelming level of workloads etc., this causes the Cloud service providers to breach the agreed SLA [8]. Herein, achieving an optimised balance in the provisioned level of resources during workload execution is a long-standing objective of the service providers, in such a way to achieve energy conservation without causing workload failures. Cloud workloads are viewed in the form of jobs, every jobs may encompasses one to several number of integral tasks. Such tasks within jobs require similar or distinct server resources such as CPU cores, memory capacity, and so on. Tasks may be assigned to same or different servers for independent execution. Some of these tasks can act as potential stragglers, such that the straggling tasks runs longer and/or consume more resources than majority of the other co-located tasks within a given job. Understanding the resource requirements of the internal tasks of jobs is important, as such insights might help managing the tasks in an energy efficient way. Cloud provides usually record the usage profiles of the workload, analysing such usage profiles can provide with such understanding of the task behaviours.

In this context, forecasting the future behaviours of the tasks are one of the possible ways of achieving energy efficient workload execution in Cloud datacentres without affecting the execution performance. Developing a prediction model naturally requires characterisation of the Cloud workloads through descriptive analytics of the jobs and tasks. But such analytics of Cloud workloads can often be complex due to the heterogeneous and dynamic nature of the Cloud workloads, since the workloads processed in a single datacentre arrive from various business context. Despite the existing works of workload characterisation and prediction modelling [9], [10], Cloud Computing still requires an effective analytics of workloads to sufficiently exploit the features and relationships between Cloud users and workloads to develop a reliable prediction model. To this end, this paper proposes an intelligent prediction framework based on big data analysis and artificial intelligence, aimed at predicting the resource requests of Cloud worloads over a defined period of time in the future. Our proposed prediction framework exploits an improved GRU neural network integrated with a novel resource stragglers detection technique and achieves a reliable level of accuracy whilst predicting the resource requests. Important contributions of this paper include the following:

1) Analysis and extraction of predictive characteristics of Cloud workloads to build the predictability profiles of task resource requests. The periodicity characteristics of Cloud workloads that can aid prediction are exhibited.

2) A straggler detection technique has been proposed for task-level resource requests, which exploits anallytics of both the historical task resource usage data and task event data.

3) An improved GRU network model based on RNN has been used to handle time series workload data. This model can not only capture the relationships between data on long term time sequential sequences, but can also effectively disregard irrelevant information of workload data.

4) An efficiently integrated prediction framework named IGRU-SD consisting of NS-2R and S-2R prediction models has been developed to predict the task resource requests based on the periodicity characteristics of Cloud workloads, namely the hour-of-day and day-of-week patterns. The proposed NS-2R and S-2R prediction models are used to forecast and classify the non-straggler tasks and straggler tasks among the workload data.

The rest of the paper is organised as follows: Section II reviews the related works. Section III presents the required background on Cloud workloads. Section IV describes the proposed prediction IGRU-SD framework including the improved GRU prediction model and the detection method of resource requests stragglers. Section V introduces the experimental data sets, parameter set and evaluation index. In Section VI, the performance of our proposed prediction framework is discussed by comparing with other chosen state-of-the-art prediction models. Section VII concludes the paper, along with outlining our future research directions.

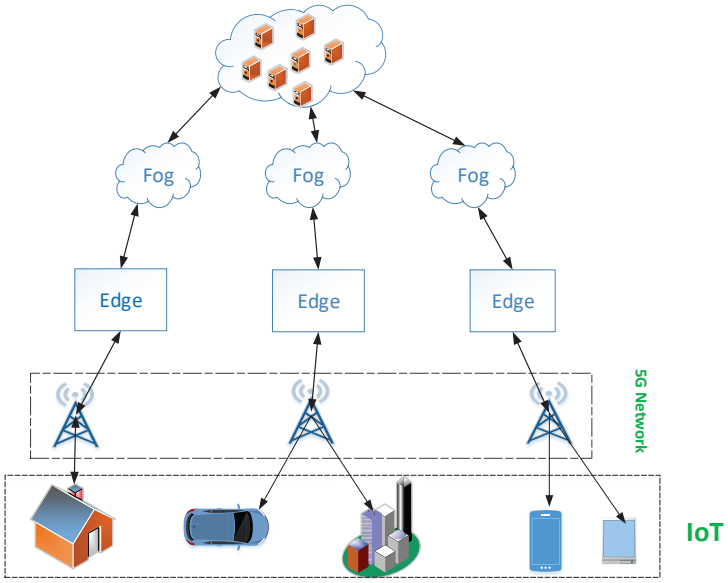


Fig. 1. Hybrid Intelligent Network Architecture

II. Related Work

Resource management in cloud datacentres such as resource prediction, resource scheduling, task allocation, etc. is being widely researched in the recent years. The literature [11]uses a variety of realistic policies and realistic test scenarios to analyse the impact of virtual machine allocation on resource consumption. This work have claimed that the total resource and energy usage can be reduced through the deployment of special allocation policies within various virtual machines. . A resource management scheme named PLAN [12] has been proposed, which is a policy-aware and network-aware VM (Virtual Machine) management scheme aimed at reducing the communication costs incurred during VM migration in Cloud data centres whilst meeting network policy requirements. This issue has been modelled as an NP-hard problem, PLAN deployed an effective algorithm to reduce the communication cost while satisfying the policy constraints.

The predictability of resource-intensive computing workloads in large-scale computing data centres has been analysed in the literature [13], which firstly introduced the problem statement of resource interference in Cloud data centres, especially that the resource-intensive computation may need more CPU resources, such as big data analysis processing. According to their problem statement, a predictability model has been proposed with calculation parameters and the environmental parameters that are identified to be obviously affecting the execution behaviours.

In order to address the challenging issues of resource optimisation in Cloud data centres, [14] proposed a Cloud resource scheme integrated with a novel resource prediction model based on GA (Genetic Algorithm) and a VM placement algorithm for improving the proportion of resource utilisation and reducing energy consumption. This work modelled the VM provisioning as a MOO (Multi Objects Optimisation) problem and addressed it with a GA approach. This applied GA prediction model has obtained an optimal solution through its crossover operation, and prevents the model from falling to a regional optimal solution through a mutation operation.

Cloud workloads are becoming more heterogeneous. This inherent task heterogeneity increases the complexity of resource management for Cloud providers. There is a phenomenon in Cloud data centres, where a large number of resource-intensive tasks is concentrated within a certain period of time. In order to meet the resource requirements of such tasks, Cloud service providers have to provide a mass of resources for a prolonged time period. This usually results in the resource idleness issue, where by the majority of the allocated mass resources are not even utilised during the actual execution of tasks. To this end, a novel analytics model [15] has been proposed, which integrates provisions to estimate the resource requirements of tasks before scheduling, to classify stragglers and non-stragglers, and attempts to optimise the resource provisioning in such a way that the resource idleness are reduced.

In the recent years, with the breakthrough and development of AI (Artificial Intelligence) technology, the ways of achieving smart resource management are shifting directions. Researchers are attempting to exploit AI to optimise resource management in data centres [11] [16]–[18]. A LSTM neural network model has been used to predict the arrival number of jobs to schedule computational jobs in the works of [16]and [17].An improved N-LSTM [18] has been proposed to predict the anticipated amounts of workloads at the VM-level within a short-term. A latency-based prediction approach has been proposed, which integrates the K-means clustering algorithm with an improved BP neural network [11] with the intention of forecasting the task-level CPU and memory consumption levels. Compared with the traditional statistical methods, AI models are more adept at capturing the internal characteristics of Cloud workload data. In many cases, AI-related prediction models exhibit higher efficiency than the statistical models. Therefore, this paper consideres the use of AI technology to manage resources in Cloud datacentres as the future way forward in the context of smart resource and workload management.

III. Background

A. Cloud Users

We are witnessing an ever increasing trend of both the Cloud users and service providers. Although users and providers belong to various business context, the ways of processing and managing their workloads in the datacentres are often very similar. Users are usually required to complete a registration process before using the Cloud services, and will go through an authentication process every time they access the Cloud resources. Once this process is completed successfully, users can submit their workload requests to the datacentres, such requests are usually received and processed by a broker or a scheduler. In Cloud datacentres, job submissions are typically characterised by associated user IDs, assigned logical names, and corresponding resource requirements [7]. User IDs are uniquely assigned to individual users and are not duplicated. But it might be possible for a single user to sign up for different user IDs. Jobs originating from a single user but in different user ID, might characterise some level of similarity in terms of the configuration, resource requirements, latency requirements etc. It is a common belief that user behaviours or requirements might not characterise a higher degree of deviation among their submitted jobs. From a wider perspective, Cloud jobs submitted by similar user groups are often similar, users in such groups may engage in similar work life interests, and the Cloud jobs are usually the digital response of corresponding business behaviours in Cloud datacentres. In addition, the behavioural characteristics of Cloud users might be different at different time periods within a day. In other words, Cloud jobs are often associated with their operating business hours, thus exhibit weekly/week-end trends [19]. Thus, characterising user behaviours over time might provide us with useful inferences in terms of their arrival frequency, computational intensity, workload volume etc.

B. Cloud Workloads

Cloud workloads often appear in Cloud datacentres in the form of jobs or tasks characterised by various inherent attributes. The duration of job execution and the number of tasks in a job are two important indicators that define workload characteristics. The job execution duration [19], [20]is usually bimodal, so that tasks contained within a given job are either shorter and/or longer in terms of their execution time. Long-running tasks can be further classified as interactive and computationally intensive workloads. While the former requires frequent interaction with users to during execution, the latter usually refers to the processing of weblogs. Tasks with shorter execution times can be further classified into shorter CPU intensive tasks and shorter memory intensive tasks. Through the analysis of a large amount of historical workload data, Cloud jobs have been identified as completing their execution within 15 minutes, with the execution duration of a very insignificant proportions of jobs exceeding 300 minutes [21]. Task duration depends largely on job characteristics driven by the behavioural characteristics of corresponding users. In general, a single job may contain both short-cycle and long-cycle tasks. Long-cycle tasks typically hog server resources

for a long time, which means that such tasks consume more energy. It is worth noting that some short-cycle tasks may also consume a lot of energy within a shorter period of time, because they occupy a lot of resources. Most of the jobs arriving at a typical Cloud datacentre characterise dozens to hundreds of tasks, while a very few proportions of jobs processed in a datacentre characterise just a single task. It is worth noting that a few jobs have been identified to characterise more than 2,000 tasks [7]. The completion time of the entire job is determined by the completion time of all of its integral tasks. For Cloud tasks executed in parallel, the time of job completion is determined by the task with the longest execution time. For sequential tasks, the completion of the last task in the sequence is usually regarded as the completion of the corresponding job. In either case, the size, volume and duration of each individual task plays a vital role in the completion of the entire job.

C. Profiles of Cloud Users, Jobs and Tasks

1) User Profile: Jobs arriving at the datacentres accompany specific information to describe their job profile, which is a composite consisting of submission time t_s, user name n_u, and the logical job name n_j, as shown in Equation 1. The specific information describing a task are stored in the task profile, which is a composite consisting of submission time t_s, user name n_u and the corresponding resource request for CPU cores r_c and resource request for memory r_m, as shown in Equation 2.

$$J = \{t_s, n_u, n_j\} \tag{1}$$

$$T = \{t_s, n_u, r_c, r_m\} \tag{2}$$

Job and task profiles typically contain submission times and user names, which are used to build the user profiles. Therefore, Cloud user profile can be defined as composite U, comprising job submission time t_s, user name n_u, job name n_j, and their related resource requirements in terms of CPU r_c and memory r_m, as shown in Equation 3.

$$U = \{t_s, n_u, n_j, r_c, r_m\} \tag{3}$$

2) Job Event Profile: The job event profile can be defined as a composite E_j, consisting of the timestamp of submission t_s, a job ID J_i, a job name J_n, the number of encompassed tasks n_t, resource levels $c_{(c,m)}$and job scheduling class J_{sh}, as shown in Equation 4.

$$E_j = \{t_{sj}, J_i, J_n, n_t, c_{(c,m)}, J_{sh}\} \tag{4}$$

3) Task Event Profile: Since tasks within jobs are processed individually, execution profiles of the tasks encompassed within jobs can be defined as a composite E_t [19], consisting of the timestamp of submission t_{st}, task index within a given job T_i, job ID J_i to which the task belongs, resource request for CPU cores rc and resource request for memory rm of the ith task within the job J_i and the task priority T_p, as shown in Equation 5, where, the tasks with larger priorities generally have preference for resources over tasks with low priorities.

$$E_t = \{t_{st}, T_i, J_i, r_{ci}, r_{mi}, T_p\} \tag{5}$$

4) Task Resource Usage Profile: Tasks are usually executed in execution instances such as VMs or containers, and the resources allocated to individual instances might be isolated from each other. When a task is executed, related execution information is collected to form a resource usage file. A task resource usage profile can be described as a composite consisting of a start time t_{sh}, a job ID J_i to which the task belongs, task index t_i, CPU usage r_{ci}, memory usage r_{mi}, as shown Equation 6.

$$E'_t = \{t_{sh}, J_i, t_i, r_{ci}, r_{mi}\} \tag{6}$$

According to the task priority, resource requirements of CPU and memory, the task execution time t_j can be calculated in an execution instance (e.g. a Linux container) using the task completion time t_f and the time of task scheduling t_{sh}, as shown in Equation 7.

$$l_j = t_f - t_{sh} \tag{7}$$

The datasets used in this paper mainly are captured from Google Cloud computing datacentres. In Google's resource usage profiles, the typical sample measurement cycle is 5 minutes (300s) [20]. Since the completion time of each task is different and may span across several sampling cycles, further calculations are required to finally determine the execution time and resource consumption of each task. The total amount of resources consumed by a single task and its execution duration are given by the sum of all measurement samples for the corresponding task, as shown Equation 8 and 9,

$$R_T = \sum_{i=1}^{n} r_i \tag{8}$$

$$L_T = \sum_{i=1}^{n} l_i \tag{9}$$

where R_T and L_T are the total amount of resources consumed and task length for task T respectively, n is the total number of measurement samples, r_i and l_i are the resource consumption and the duration of the corresponding task execution during the ith sample period of task T. It is obvious that a given task T would require a minimal level of R_T resources and can be expected to run for a minimum duration of L_T when provisioned with R_T.

D. Patterns of Cloud Resource Requests

Previously researches [22], [23]have empirically exhibited the trends of job arrival frequency and user activities, proving such trend to be driven by business behaviours. This is to say that the resource consumption trends characterise periodicities, such as hour-of-day pattern and day-of-week pattern etc. The extraction of these features can help predicting the user intentions and resource consumption levels within a specific period of time, which can aid Cloud managers with better management of resources in order to save their operating costs. However, resource consumption of Cloud workloads changes dynamically over time, implying it is difficult to accurately capture such highly varying dynamic characteristics using traditional prediction models. If resources are allocated

solely based on the projected resource consumption levels, even minor prediction errors can cause irreparable disasters in a large-scale Cloud datacentres. Moreover, incorporating the task volume in the task profile and configuration file can provide inferring the task behaviours better, even before the start of the execution.

Fig. 2 and Fig. 3 exhibit the periodicity trends in terms of the resource requests, starting from D1 which is a Monday. Fig. 2 shows the total number of CPU resource requests received by the studied datacentre each day over a three-week period. It can be seen from Fig. 2 that the total number of resource requests received is higher during the weekdays and significantly reducing over the weekends. This also conforms to the behaviour characteristics of users in real life discussed above. In addition, corresponding days of the week over the observational period are exhibiting nearly similar level of received resource requests.

Fig. 3 illustrates the hour-of-the-day pattern of CPU resource requests. In Fig. 3, CPU resource requests are aggregated in hours, where a total of 504 hours of resource requests are analysed across a period of 21 days. It is obvious from Fig. 3 that the total amount of resource requests shows a gradual increasing as the day progresses, and deceasing towards the close of the business hours. Moreover, this pattern is observed to be cyclical over the observed period of 21.

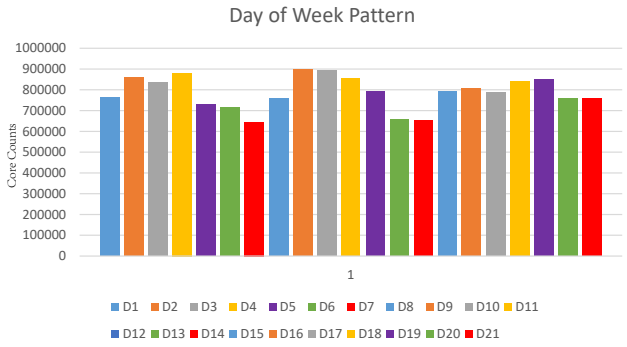


Fig. 2. CPU Resource Requests of Days

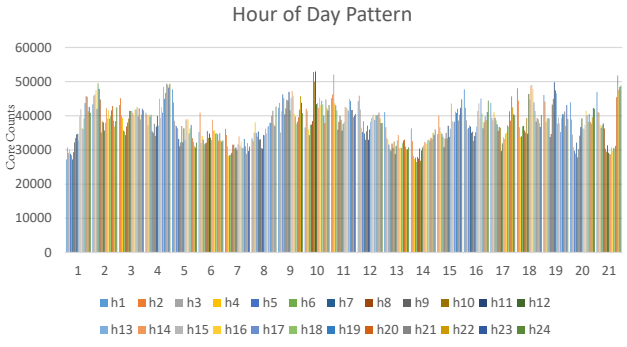


Fig. 3. CPU Resource Requests of Hours

IV. THE PROPOSED PREDICTION FRAMEWORK

This section describes our proposed prediction framework focused on predicting the workload resource requests at Cloud datacentres.

A. Model Architecture

The proposed prediction framework is illustrated in Fig. 4, which encompasses various components for functionalities such as workload data input, stragglers detection, resource requests prediction, and prediction results output. The primary purposes of the encompassed functionalities are detailed as follows.

Workload data input: The primary objective of this module is to input the resource consumption data set of tasks extracted from task resource usage trace logs. The resource consumption levels including CPU and memory of the tasks under every sample period is recorded in the task resource usage file.

Stragglers Detection: Based on the recorded resource consumption levels of tasks, every task within a given job will be subjected to a straggler classification module. The methodology used in this straggler classification module is detailed in section 4.c.

NS-2R model: Based on the stragglers detection outcome, this NS-2R model will forecast the future resource requests of tasks detected as non-stragglers for a defined time period based on the RNN-GRU model.

S-2R model: This is a prediction model designed for forecasting task-level resource request for tasks detected as stragglers. Based on the outcome of the straggler detection module, tasks labelled as resource consumption stragglers are filtered out from the task resource usage files. The task index parameter in the resource usage files can acts as an indicator of straggler tasks, which usually characterise higher CPU or memory requirements compared to the remaining non-straggler tasks within the same job. Based on this straggler detection module, this S-2R model will forecast the future resource requests of straggler tasks for a defined time period based on the RNN-GRU model.

B. Recurrent Neural Network

The emergence of RNN [24](Recurrent Neural Network) is to compensate for the shortcomings of feedforward neural networks. It is unique in such a way that RNN can use historical time-series data to guide current decisions. The data points within a given time-series are not independent, and are related to the corresponding data points in the historical time period. Failure to capture these inherent time characteristics would considerably affect the model accuracy and reliability. Cloud workload dataset characterise obvious time characteristics, which is evident from the periodicity characteristics discusses earlier. During the prediction process, it is not only important to capture the periodicity characteristics of the workloads but also the model should extract and exploit the way that the data points are associated with time. A prediction model with such a capability would deliver reliable forecast of the resource requirements of Cloud workloads. Traditional NN (Neural Networks) models can only reflect the mapping relationship between data points, but they cannot extract and utilise the time relationship between data points, which limits their applicability for accurately predicting Cloud workload

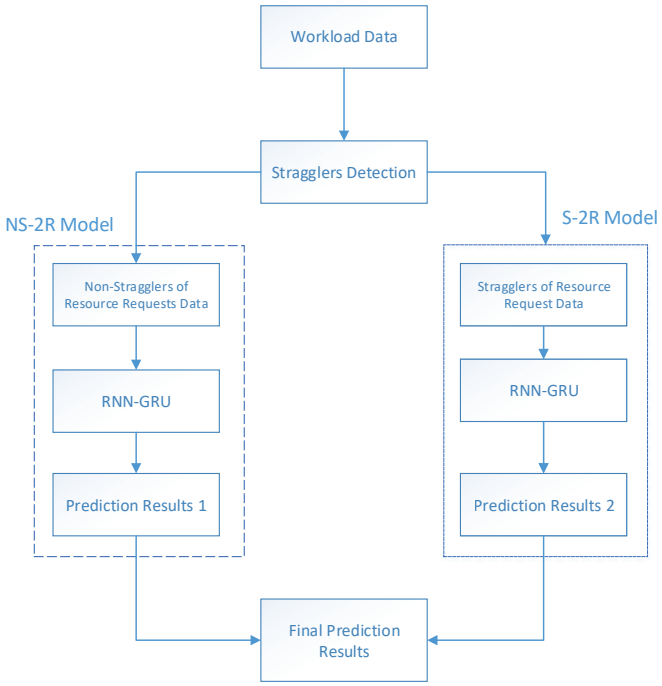


Fig. 4. The Model Architecture of the Proposed Prediction Framework

datasets. Therefore, in this paper, the resource requirements of future workloads are predicted based on RNN which often exhibit good processing capability for time-series datasets.

The basic idea principle ofbehind RNN is to use the self-feedback neuron to memorisze historical information and apply historical information to compute the data point values in the calculation of the next instance moment. The detailed calculation process will be explained in Figs. 5 and Fig. 6 presents a detailed overview of the computation process of RNN. Fig. 5 depicts a typical three-layer RNN model structure which is composed of anthe input layer, athe hidden layer and thean output layer. Each layer contains a number of neurons and full a complete connection between each layer is established throughand the neurons, . There are different connection based on various degrees of connection weights between the layers. Where In Fig. 5 and Fig. 6, W_U represents a connection weight matrix between the input layer and the hidden layer, and W_V represents a connection weight matrix between the hidden layer and the output layer, this structure resembles a traditional neural network. It is worth noting that RNN has a special structure to that of the traditional neural network, by including a self-loop structure in the hidden layer. W_H represents the connection weight matrix of the hidden layer, flowing from the previous hidden layer to the current hidden layer in time. Because of the fact the weights of neurons are shared at each moment in time period, the expanded RNN can be regarded as a multi-layer feedforward neural network shared by the involved parameters. This feature of RNN plays a very important role in processing time-series data. Firstly, the parameter sharing mechanism enables RNN to data mine key information in the time-series without being affected by the information position. Secondly, the parameter sharing mechanism enables RNN to process data sequences of different lengths.

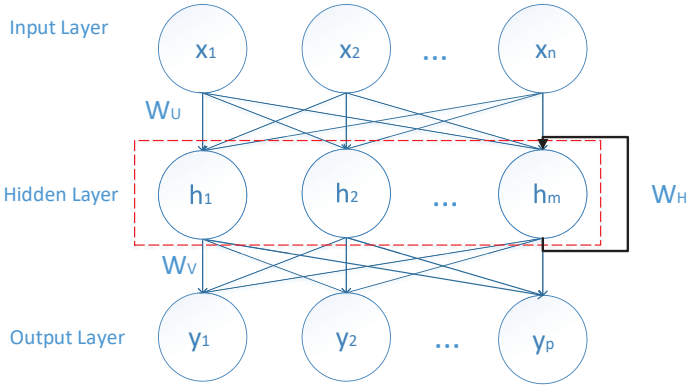


Fig. 5. The Structure of RNN

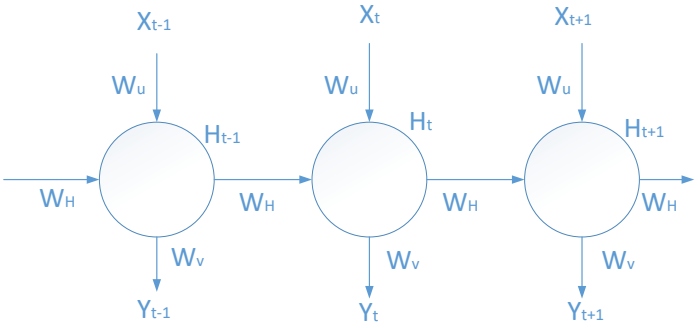


Fig. 6. The Basic Recurrent Unit

The working mechanism of RNN is described as follows. The input sequence of RNN is represented by an x sequence of length t, so that $X = \{X_1, X_2, ..., X_t, X_{t+1}, ..., X_T\}$, where $X_t = (x_1, x_2, ..., x_n)$) indicating that X has n number of input values at time t. This is to say the dimension of X is n at time t. For the same reason, the value of the hidden layer can be expressed as $H_t = (h_1, h_2, \ldots, h_m)$, where m is the number of neurons in the hidden layer. The output of RNN can be represented with a composition $Y_t = (y_1, y_2, \ldots, y_p)$, where p is the number of outputs at the time t. According to this definition, H_t can be further computed using the following equations.

$$Z_t = W_u * X_t + W_H * H_{t-1} + a \tag{10}$$

$$H_t = f(Z_t) \tag{11}$$

where, Z_t is the input value of the neurons in the hidden layer, a is an offset vector of the hidden layer, $f()$ is an activation function of the hidden layer. From this, the value of Y at time t can be calculated using the following equation.

$$Y_t = g(W_v * H_t + b) \tag{12}$$

where, $g()$ is the activation function of the neurons in the output layer, b is the offset vector of the output layer.

The ultimate goal of RNN is to minimise the objective function. In this process, RNN uses the BPTT (Backpropagation Through Time) algorithm to compute the gradients, as follows.

$$J = \sum_{t=1}^{T} J_t \tag{13}$$

$$J_t = -Y_t log Y_t \tag{14}$$

$$J = -\sum_{t=1}^{T} Y_t log Y_t \tag{15}$$

(1) The process of computing the gradient of J respect to W_V is listed as belows:

$$\frac{\partial J}{\partial W_V} = \frac{\partial \sum_{t=1}^{T} J_t}{\partial W_V} = \sum_{t=1}^{T} \frac{\partial J_t}{\partial W_V} \tag{16}$$

$$Y_t = g(Q_t) \tag{17}$$

$$Q_t = W_v H_t + b \tag{18}$$

$$\frac{\partial J_t}{\partial W_V} = \frac{\partial J_t}{\partial Y_t}\frac{\partial Y_t}{\partial W_V} = \frac{\partial J_t}{\partial Y_t}\frac{\partial Y_t}{\partial Q_t}\frac{\partial Q_t}{\partial W_V} \tag{19}$$

(2) The process of computing the gradient of J respect to W_H is listed as belows:

$$\frac{\partial J}{\partial W_H} = \frac{\partial \sum_{t=1}^{T} \frac{\partial J_t}{\partial W_H}}{\partial W_H} = \sum_{t=1}^{T} \frac{\partial H_t}{\partial W_H}\frac{\partial J_t}{\partial H_t} \tag{20}$$

where, H_t is a function of W_H and H_{t-1}, and H_{t-1} is a function of W_H and H_{t-2}. Using the chain rule, the following equations are obtained:

$$\frac{\partial J}{\partial W_H} = \sum_{t=1}^{T}\sum_{k=1}^{t} \frac{\partial H_k}{\partial W_H}\frac{\partial H_t}{\partial H_k}\frac{\partial Y_t}{\partial H_t}\frac{\partial J_t}{\partial Y_t} \tag{21}$$

$$H_t = f(W_H H_{t-1} + W_U X_t + a) \tag{22}$$

(3) The process of computing the gradient of J respect to W_U is listed as belows:

$$\frac{\partial J}{\partial W_U} = \frac{\partial \sum_{t=1}^{T} J_t}{\partial W_U} = \sum_{t=1}^{T} \frac{\partial J_t}{\partial W_U} = \sum_{t=1}^{T} \frac{\partial H_t}{\partial W_U}\frac{\partial J_t}{\partial H_t} \tag{23}$$

Using the chain rule, the following equation is obtained:

$$\frac{\partial J}{\partial W_U} = \sum_{t=1}^{T}\sum_{k=1}^{t} \frac{\partial H_k}{\partial W_U}\frac{\partial H_t}{\partial H_k}\frac{\partial Y_t}{\partial H_t}\frac{\partial J_t}{\partial Y_t} \tag{24}$$

where,

$$\frac{\partial H_t}{\partial H_k} = \prod_{i=k+1}^{t} \frac{\partial H_i}{\partial H_{i-1}} = \prod_{i=k+1}^{t} W_H^T diag[f'(t-1)] \tag{25}$$

The gradient of W_U is:

$$\frac{\partial J}{\partial W_U} = \sum_{t=1}^{T}\sum_{k=1}^{t} \frac{\partial H_k}{\partial W_U}\left(\prod_{i=k+1}^{t} W_H^T diag[f'(t-1)]\right)\frac{\partial Y_t}{\partial H_t}\frac{\partial J_t}{\partial Y_t} \tag{26}$$

If we define $\gamma = \left\|W_H^T diag[f'(t-1)]\right\|$, the expression in parentheses above can be expressed as γ^{t-k}. if $\gamma > 1$, while $t-k \to \infty$, then $\gamma^{t-k} \to \infty$. In this case, the gradient explosion phenomenon will occur in the RNN model. If $\gamma < 1$, while $t-k \to \infty$, $\gamma^{t-k} \to 0$. In this case, the gradient disappearance phenomenon will occur in the RNN model.

C. Improved GRU Based on RNN

The RNN model described above is a neural network model specifically designed to process time-series datasets, which can better fit the nonlinear relationship between the data points in a time coordinate. However, RNN has an obvious shortcoming that its gradient can disappear or explode easily, especially for long time-series datasets, during the process of model training. The objective of the proposed GRU (Gate Recurrent Unit) is to alleviate the phenomenon of gradient disappearance that is widespread in RNN. GRU is a logic gate control unit, and its structure is shown in Fig. 7 [25]. In order to solve the gradient disappearance problem prevailing in the RNN model, GRU is used to replace the hidden layer loop structure in the classical RNN model.

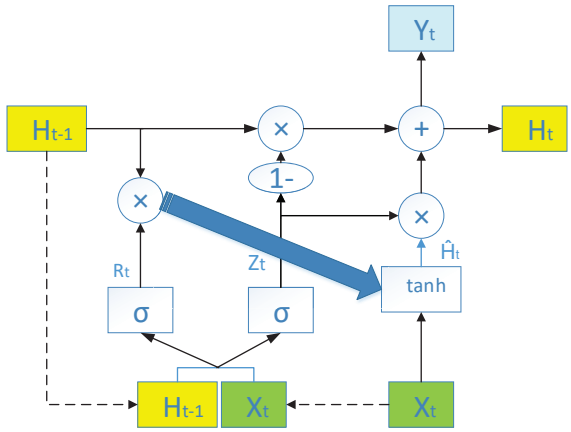


Fig. 7. The Structure of GRU

The input and output structure of GRU shown in Fig.7 is exactly the same as the RNN model , where X_t is the input, and H_{t-1} is the hidden state obtained from the previous node. This hidden state contains information of the previous node. GRU is regarded superior to RNN such that the former can control the states of information flow in the neural network at different time steps. Having Computing X_t and H_{t-1}, the output value Y_t of the current node and the hidden state h_t are passed to the next node. GRU model has two unique gate control units, namely the update gate unit Z_t and the reset gate unit R_t. The update gate is used to obtain a balance between the historical memory information and the current time step input information. With smaller values of update gate, the model can precisely focus on the information of the previous hidden layer state; otherwise, the model is more focused on the current information. The function of the reset gate is to forget parts of the hidden layer information at the previous moment, and the amount of information to be forgotten is determined by the value of the reset gate. A smaller value in the reset gate depicts more information to be forgotten, so that less historical information are introduced, and vice versa. The values of the

two gates are all determined by the previous hidden layer state H_t and the input of current time step X_t, as Equations 27 and 28.

$$R_t = \sigma(W_r \otimes [H_{t-1}, X_t]) \tag{27}$$

$$Z_t = \sigma(W_z \otimes [H_{t-1}, X_t]) \tag{28}$$

where, W_r and W_z are the weights of the reset gate and update gate respectively, σ is an activation function. The symbol $\otimes$ denotes the matrix multiplication.

By resetting the gate R_t, the candidate vector H_t that needs to be added to the hidden state of the current time step is calculated, which is shown in Equation 5, where, the symbol * expresses multiplication of the corresponding elements in the two vectors. $Tanh()$ is an activation function which can scale data to a range of -1 to 1. $\hat{H}_t$mainly contains the current input data of X_t. H_t is added to the current hidden state in a targeted manner, which is equivalent to "memorising the state of the current moment."

$$\hat{H}_t = tanh(W \bullet [R_t * H_{t-1}, X_t]) \tag{29}$$

The next stage is to update the memory, which is also the most critical step. This stage involves two different computation process, such as “remembering” and “forgetting”. The value of the update gate Z_t and $\hat{H}_t$ will be used in this stage. Now an update function is obtained as shown in Equation 30.

$$H_t = (1 - Z_t) * H_{t-1} + Z_t * \hat{H}_t \tag{30}$$

In this equation, $(1 - Z_t) * H_{t-1}$ indicates the selective "forgetting" nature of the original hidden state. $(1 - Z_t)$ here can be imagined as a forget gate, which is used to forget some unimportant information in the dimension of H_{t-1}. $Z_t * \hat{H}_t$ indicates the selective "remembering" nature of $\hat{H}_t$, which contains the current node information. Similar to the above process, $Z_t * \hat{H}_t$ also forgets some unimportant information in $\hat{H}_t$ dimension. This process can also be described as the way of choosing the most relevant information in the $\hat{H}_t$ dimension. It is worth noting that the gating signal Z_t ranges from 0 to 1. When the value of Z_t is close to 1, less information will be remembered; when it is close to 0, less information will be forgotten. During this process, all the parameters and the weights of the gate control unit are trained by the back propagation algorithm.

D. Stragglers Auto-Detection Module

1) Resource Hunger stragglers: **Definition (Resource Hunger Stragglers):**For a job $J = \{t_1, t_2, t_3 ... t_n\}$, where n is the total number of tasks contained in the job J, the average CPU usage rate of these tasks is μ, and the average task length is ω, then tasks which exhibit both higher CPU usage rate thanμ and a runtime duration longer thanω are termed as resource hunger stragglers in the job J.

A Cloud job usually contains a certain number of tasks, and the number of tasks may vary from job to job. The level of resource and execution time consumed by different tasks within a job may also vary. Some of these tasks have a significantly higher CPU usage rate and execution time than others within the same job. Therefore, if the CPU usage rate and execution time of a task are higher than the average CPU usage rate and execution time of all tasks within this job, such task is generally recognised as a resource hunger or energy-aware straggler, as shown in Equation 31.

$$S_t[i] = (U_C[i] > \mu) \cap (L_T[i]) > \omega) \tag{31}$$

where, S_t presents energy-aware stragglers, U_C is the CPU usage rate of task i, L_T is the length of the ith task, and μ is the average tasks’ CPU usage rate and ω is the average duration of the tasks contained in the respective job.

The number of resource hunger stragglers varies from job to job. Although it is ideal to use a fixed linear expression to describe the proportional relationship between the number of resource hunger stragglers and the total number of tasks in a job, this relationship is often non-linear. In our former work [19], 5 jobs with different number of tasks have been randomly selected to analyse the proportional presence of stragglers, the results of this study are presented in Fig. 8. The number of tasks contained in the selected jobs are 50, 100, 200, 488 and 1050 for job1, job2, job3, job4 and job5 respectively. From Fig. 8, job1 contains 8% of resource hunger stragglers. Job5 contains a very small proportion of resource hunger stragglers, just around at 2%. The proportions of resource hunger stragglers within job2 and job4 are about 27% and 36%, respectively. In addition, more than 40% of the tasks in job 4 have been identified as resource hunger stragglers, which is a relatively large proportion among these studied jobs. Such results have empirically proved the non-linear relationship existing between the number of energy-aware stragglers and the total number of tasks within a given job.

Task-level resource hunger stragglers are usually difficult to be identified, since their actual causes are often implicit. Due to the dynamicity of Cloud jobs in terms of resource requirements, tasks that represent higher CPU or memory requirements can be potential resource hunger stragglers during runtime compared with the remaining tasks within the same job. It is worth noting that in some cases task-level resource stragglers are inevitably driven by the actual task requirements.

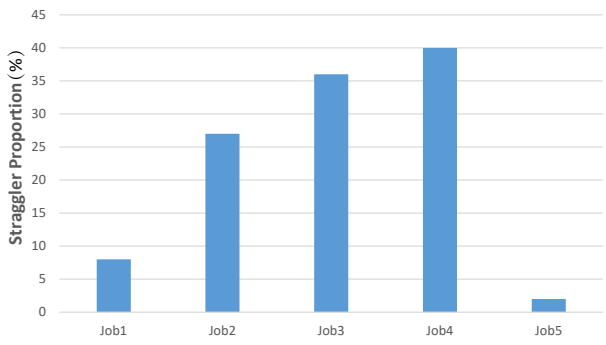


Fig. 8. Straggler Proportions in Selected Jobs

2) Long Tail stragglers: **Definition (long tail stragglers):** For a job $J = \{t_1, t_2, t_3 ... t_n\}$, where n is the total number of tasks contained in the job J, the tasks which exhibit a runtime duration as an increased multiple of a majority of the remaining tasks within the same job are termed as long-tail stragglers [19]in the job J.

Long-tail stragglers are often only witnessed in a small part of the total tasks in a given job, but they will occupy server resources for a long time, which means such tasks need to consume more resources than the other tasks under same CPU frequency. There are two ways to run tasks, which are serial computing and parallel computing. Long-tail tasks in the serial mode have less impact on the overall completion time of a job, but long-tail tasks in the parallel operation play a key role in the final completion of a job. Most of the existing literature uses a threshold of 50% [26] to identify long tails, whereby a task exhibiting a runtime duration 50% more than the average runtime duration of all tasks within the same job is classified as a long-tail straggler, as shown in Equation 32.

$$d[i] > davg * 1.5 \tag{32}$$

where, $d[i]$ is the runtime duration of task i contained in a job, d_{avg} is the average runtime duration of all the tasks within the same job.

V. EXPERIMENT TESTS

A. Dataset

The dataset used in this paper was released by Google, which is a large-scale trace logs collected from Google Cloud Data Centre, which includes 5320 users, 12500 operating servers, 650892 Cloud job submissions and 46093201 task submissions, spanning across 29 days [27]. It is worth noting that the sampling period of the data is 300s, which means the resource monitor performs a sampling operation on the running server every 300 seconds. Multiple trace logs will be recorded for each sample. The specific statistical information about the dataset is presented in Table 1. However, only the first 21 days of data from day 1 to day 21 are used in our experiments due to data ambiguity and missing information in the remaining 7 days. Moreover, the dataset is split into weeks, in order to analyse and exploit the day-of-the-week pattern existing in the workloads. Due to the heterogeneous distribution of the resource request dataset, all the sample data is normalised with the following equation before training the dataset in our proposed RNN-GRU model. Further, a reverse normalisation operation is performed before the output phase, so that the output result can be mapped to the original data range. The purpose of this is to achieve a similar data distribution among the input sample, so that the model can fit the target better and converge faster.

$$x'_t = \frac{x_t - x_{min}}{x_{max} - x_{min}} \tag{33}$$

TABLE I
OVERVIEW OF THE GOOGLE DATASET

Size of the Dataset	41G
Timespan	29 days
Number of Users	5320
Number of Operating Servers	12500
Number of Job Submissions	650892
Number of Task Submissions	46093201

B. Experimental Parameters Setting

The hardware environment used in the experiments including the following: CPU: Intel Core i5-8500 @3.00 Hz, 8G Memory. The simulation processes are mainly performed on MATLAB R2018a.

The RNN-GRU model uses a three-layer network structure, and the input-output sequence operates on a multiple input to single output mode. The number of hidden layer neurons is set to 8, the initial learning rate is set to 0.0001, the batch dimension is 1, and each test is repeated 10 times in order to reduce random errors. Other parameter settings of the experiments are introduced in the following sections, where appropriate.

C. Evaluation Metric

For demonstrating the prediction efficiency, the effectiveness and reliability of the prediction models need to be systematically evaluated through scientific evaluation methods. Based on the characteristics of our proposed model and the experiment dataset, the following evaluation metric is used in our performance evaluation.

MAPE (Mean Absolute Percentage Error): It is the mean percentage error between the predicted value and the real value. MAPE is commonly used in regression problems and model evaluation. Meanwhile, MAPE is an intuitive method to present relative errors, evaluating the accuracy based on Equation 34.

$$MAPE = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{y_t - y'_t}{y_t} \right| \tag{34}$$

where, y_t is the predicted value of user resource requests, y'_t is the real value of user resource requests, n is the total amount of data involved in the calculation.

VI. EXPERIMENTS PERFORMANCE EVALUATION

A. Prediction Performance for Hour-of-Day Pattern

According to our previous analysis presented in Section 3.4, we found that the number of requests from Cloud users for CPU resources has obvious periodicity characteristics, along with the existence of internal correlations between successive resource requests from users. Thus, an evaluation experiment has been conducted to explore the effects of user's resource requests based on the hour-of-the-day pattern. Fig. 9 presents the prediction results of our proposed framework, in estimating the user request trend. In this experiment, we accumulate the CPU resource requests received over the period of first 21 days in the dataset in chronological order by hour. This resulted in

504 sample hours, the corresponding value plotted in each time point expresses the total amount of resource requests received during that period. The first 336 hours of data are used as training samples in our proposed prediction framework, and the remaining 168 hours of data are used as test samples.

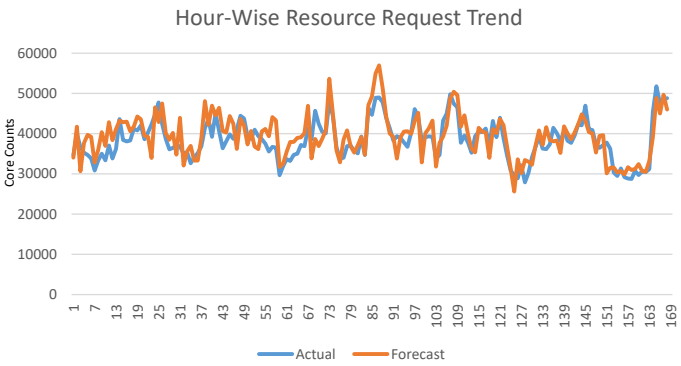


Fig. 9. Hour-Wise Resource Request Trend

Fig. 9 shows the predicted amount of resource requests against the actual resource requests trend based on our proposed method. On a coarse grain, our proposed prediction model effectively captures the trend of changes witnessed in the received CPU requests, where the peaks and valleys of CPU resource request trends are closely captured. In particular, most of the peak demands of CPU resource requests are effectively predicted, which is mainly due to the presence of the S-2R prediction module in our proposed model. We believe that most of the resource stragglers are predicted through the S-2R module, so that the final predicted output is closer to the actual value, which holds true even under situation of sudden increase in the amount of resource requests. Thus, our prediction model can maintain a high level of predictability with reliable level of accuracy. Failure the capture sudden peaks in the resource request trend might not provide sufficient insights to the providers, to help them to scale up the available level of running resources, which can significantly affect the level of QoS offered to the clients. At the same time, providing insights into the off-peak or reducing level of resource request trends helps the providers to conserve server resources through scaling down the number of operating resources. From Fig. 9, at the 85th hour, the predicted value is much higher than the actually arrived requests, this would insists the providers to scale more resources than the actual level of requirement. The reason for this anomalous point may be due to the superposition of extra resource stragglers. In simple terms, a few proportions of the non-straggler tasks are classified as straggler tasks by our model. Fortunately, such a situation is rare, so the overall impact is negligible.

Fig. 10 illustrates the prediction errors witnessed for various hour points. From Fig. 10, most of the prediction errors are less than 15%, which means that in most cases, our prediction model has a fairly high prediction accuracy, which stays consistent for a pro-longed period of time. This is because of the fact that our proposed prediction framework based on the RNN-GRU integrated with the stragglers detection module,

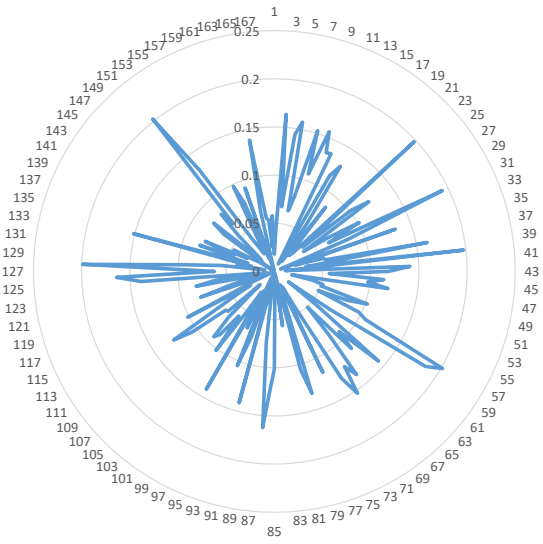


Fig. 10. Prediction Errors of Hour-Wise Resource Requests

not only captures the periodicity characteristics inherent in the time-series data, but also greatly reduces the impact of resource stragglers on prediction accuracy. In addition, only a small number of prediction errors are greater than 15%, but remain within an acceptable range of no higher than 20%. It can be concluded that our proposed NS-2R and S-2R dual prediction modules have played a significantly positive role in detecting resource stragglers and in improving the prediction accuracy of the final resource requirements.

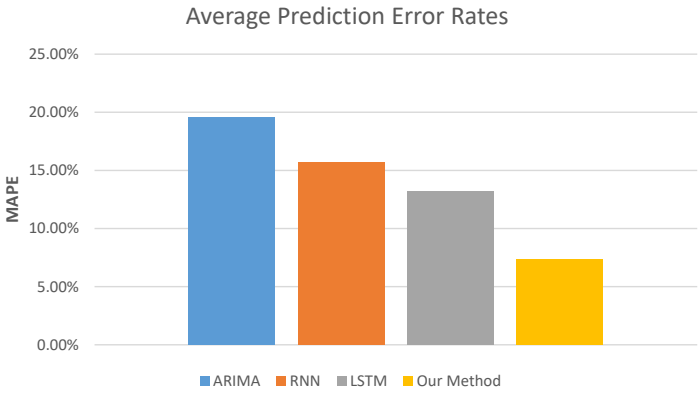


Fig. 11. Prediction Errors of Hour-Wise Resource Requests

Fig. 11 presents the average prediction error rate of our proposed model against other classical or popular prediction models such as ARIMA, RNN, and LSTM. ARIMA is a statistical model widely used for the periodic prediction of stationary stochastic processes [28]. RNN is a neural network dedicated to process time-series data. Due to its special internal structure, this model can effectively capture the relationship between the successive data points on a time-series. LSTM is a variant of the RNN model used to overcome the gradient disappearance and gradient explosion of RNN [17]. This evaluation is intended to demonstrate the efficiency

of our proposed resource requests framework based on the RNN-GRU model integrated with our novel resource stragglers detection module. From Fig. 11 it can be observed that our proposed prediction framework outperforms the other three models by exhibiting an average prediction error rate of 7.32% against ARIMA, RNN, and LSTM models, whose prediction error rate corresponds to 19.56%, 15.73% and 13.24%, respectively. By delivering a reliable level of prediction accuracy for hour-of-the-day resource requests, Cloud service providers can deploy server resources ahead of time on the basis of our proposed prediction framework.

B. Prediction Performance for Day-of-Week Pattern

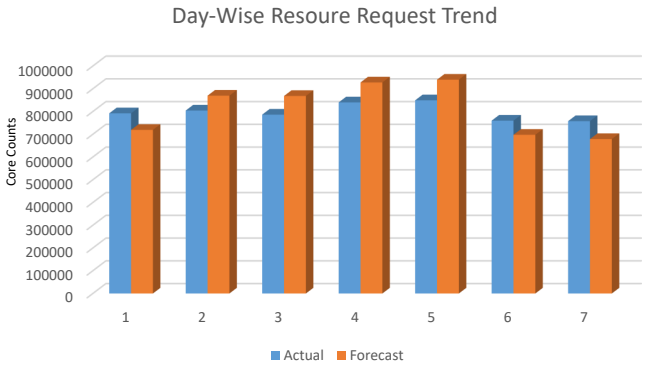


Fig. 12. Day-Wise Resource Request Trend

Fig. 12 presents the predicted and actual CPU requests of the 3rd week (day 15 to day 21). In order to evaluate the efficiency of our proposed prediction framework on estimating resource requests on a day-of-the-week pattern, the first two weeks (day 1 to day 14) of historical data are used as the training sample and the rest data are used as the test sample. In essence, the first two weeks data are used and trained in the model to predict the resource requests anticipated in the third week. On a coarse grain, the weekday increase and weekend decrease pattern in the received resource requests is evident from Fig. 12. As discussed earlier, this trend is postulated due to the operating business hours. It can be seen from Fig. 12 that our proposed prediction model effectively captures the day-of-the-week trend among user resource requests, and delivers a prediction on the expected amounts of resource requests on each day of the week with reliable level of accuracy.

Fig. 13 illustrates the prediction errors of our proposed prediction framework for each day of the week, exhibiting prediction error rates at 9.20%, 8.13%, 10.44%, 10.43%, 10.68%, 8.18%, and 10.40% for Monday through to Sunday respectively. Due to the heterogeneity of tasks with respective jobs, accurately predicting the resource requirements for each day is a huge challenge, especially during sudden increase in the resource requests. Our proposed prediction model based on RNN-GRU integrated with the stragglers detection module can control the prediction error within an acceptable range for a long term. It can be observed from Fig. 14 that all the prediction error values are below 0.11, meaning that the

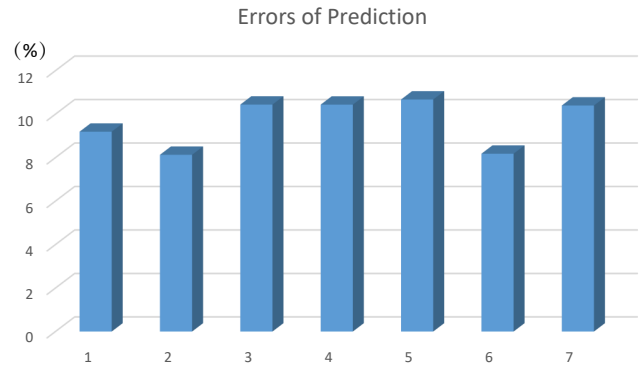


Fig. 13. Prediction error rates of Day-Wise Resource Requests

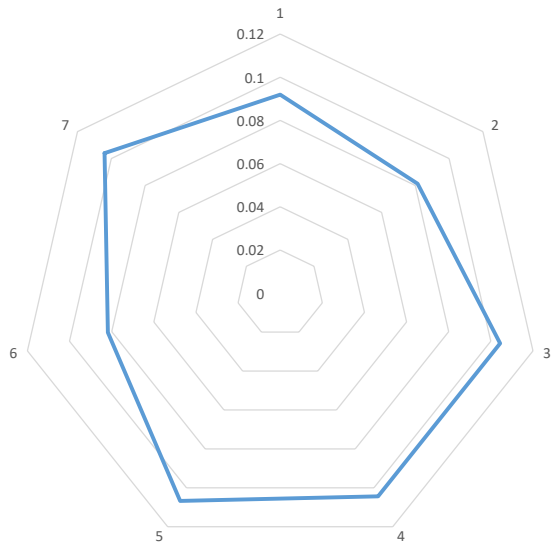


Fig. 14. Prediction Error Rates of Day-Wise Resource Requests

deviation between the predicted amount of CPU resource demands per day and the true demands is less than 11%. This is still of great significance to Cloud service providers. Cloud resource managers can develop long-term deployment strategies exploiting such resource prediction results, through shutting down redundant servers to save more resources, reducing energy consumption, and in turn reducing operating costs etc.

Fig. 15 illustrates the average prediction error rate of our proposed model for the day-of-the-week resource requests against other classical or popular prediction models such as ARIMA, RNN, and LSTM. From Fig. 16 it can be observed that our proposed prediction framework outperforms the other three models by exhibiting an average prediction error rate of 9.64% against ARIMA, RNN, and LSTM models whose error rate are witnessed at 22.81%, 18.43% and 16.68%, respectively. Due to their prolonged prediction interval, the error rates of all the compared models are high. As the time interval expands, association between the data points in the time-series becomes weaker, thus are more difficult to capture. However,

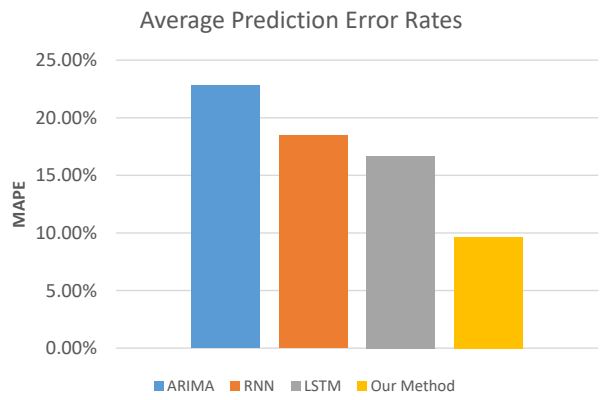


Fig. 15. Average Prediction Error Rates

our proposed prediction model still maintains a relatively low prediction error (less than 10%), which is attributed to the employment of the NS-2R and S-2R dual prediction models, so that our proposed model can still capture the inherent features of long-term time-series data, and decrease the impact of resource stragglers on the overall prediction accuracy. For large-scale Cloud service providers, even a small percentage of energy reduction can provide notable economic benefits. Besides, sufficient supply of resources is necessary to achieve a smoother execution of users' jobs. Our, proposed model can help the service providers to achieve their socio-economic objectives by delivering a reliable prediction of future resource request demands.

VII. CONCLUSION

The emergence of 5G networks are naturally expected to increase the number of IoT and mobile devices being connected to Cloud datacentres. A large number of IoT devices may require powerful compute supplements which are usually provisioned through Cloud datacentres. Therefore, resource management in large Cloud datacentres is becoming more important in the 5G era. This paper has demonstrated the benefits of prediction analytics in achieving efficient resource management in Cloud datacentres, aiding the service providers to achieve energy conservation, server management, workload execution, QoS maintenance etc., where the reliability of the prediction results plays a crucial role. This paper proposed a novel resource requests prediction framework, named IGRU-SD, based on an improved GRU neural network model and resource request stragglers detection. Experiments conducted based on real-world Cloud trace logs demonstrate that the proposed IGRU-SD framework achieves better prediction accuracy than the ARIMA, RNN and LSTM prediction models respectively. Classifying the tasks based on our proposed straggler detection module has a significantly positive effect on the prediction process. Our proposed prediction framework exhibits better prediction accuracy by predicting the resource requests of straggler tasks and non-straggler tasks over a period of time, respectively. As a future work, developing efficient resource allocation strategies for executing workloads in an energy and cost efficient way in Cloud datacentres, based on our proposed prediction framework will be explored.

ACKNOWLEDGMENT

This work was partially supported by the Natural Science Foundation of Jiangsu Province under Grant BK20170069, UK-Jiangsu 20-20 World Class University Initiative programme, and UK-Jiangsu 20-20 Initiative Pump Priming Grant. Lu Liu is the corresponding author.

REFERENCES

[1] E. Pateromichelakis, F. Moggio, C. Mannweiler, P. Arnold, M. Shariat, M. Einhaus, Q. Wei, Ö. Bulakci, and A. De Domenico, "End-to-end data analytics framework for 5g architecture," *IEEE Access*, vol. 7, pp. 40 295–40 312, 2019.

[2] L. Zhang and Y.-C. Liang, "Average throughput analysis and optimization in cooperative iot networks with short packet communication," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 11 549–11 562, 2018.

[3] S. Li, L. Da Xu, and S. Zhao, "5g internet of things: A survey," *Journal of Industrial Information Integration*, vol. 10, pp. 1–9, 2018.

[4] Z. Ning, X. Kong, F. Xia, W. Hou, and X. Wang, "Green and sustainable cloud of things: Enabling collaborative edge computing," *IEEE Communications Magazine*, vol. 57, no. 1, pp. 72–78, 2018.

[5] D. Miao, L. Liu, R. Xu, J. Panneerselvam, Y. Wu, and W. Xu, "An efficient indexing model for the fog layer of industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4487–4496, 2018.

[6] A. Kumari, S. Tanwar, S. Tyagi, N. Kumar, M. S. Obaidat, and J. J. Rodrigues, "Fog computing for smart grid systems in the 5g environment: Challenges and solutions," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 47–53, 2019.

[7] J. Panneerselvam, L. Liu, and N. Antonopoulos, "Inot-repcon: Forecasting user behavioural trend in large-scale cloud environments," *Future Generation Computer Systems*, vol. 80, pp. 322–341, 2018.

[8] B. Ciciani, D. Didona, P. Di Sanzo, R. Palmieri, S. Peluso, F. Quaglia, and P. Romano, "Automated workload characterization in cloud-based transactional data grids," in *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*. IEEE, 2012, pp. 1525–1533.

[9] J. Panneerselvam, L. Liu, N. Antonopoulos, and M. Trovati, "Latency-aware empirical analysis of the workloads for reducing excess energy consumptions at cloud datacentres," in *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*. IEEE, 2016, pp. 44–52.

[10] J. Patel, V. Jindal, I.-L. Yen, F. Bastani, J. Xu, and P. Garraghan, "Workload estimation for improving resource management decisions in the cloud," in *2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems*. IEEE, 2015, pp. 25–32.

[11] Y. Lu, L. Liu, J. Panneerselvam, X. Zhai, X. Sun, and N. Antonopoulos, "Latency-based analytic approach to forecast cloud workload trend for sustainable datacentres," *IEEE Transactions on Sustainable Computing*, 2019.

[12] L. Cui, F. P. Tso, D. P. Pezaros, and W. Jia, "Plan: a policy-aware vm management scheme for cloud data centres," in *Proceedings of the 8th International Conference on Utility and Cloud Computing*. IEEE Press, 2015, pp. 142–151.

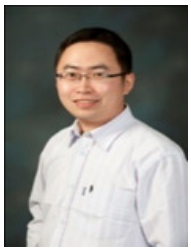
[13] C. B. Hauser, J. Domaschka, and S. Wesner, "Predictability of resource intensive big data and hpc jobs in cloud data centres," in *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 2018, pp. 358–365.

[14] F.-H. Tseng, X. Wang, L.-D. Chou, H.-C. Chao, and V. C. Leung, "Dynamic resource prediction and allocation for cloud data center using the multiobjective genetic algorithm," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1688–1699, 2017.

[15] J. Panneerselvam, L. Liu, and N. Antonopoulos, "An approach to optimise resource provision with energy-awareness in datacentres by combating task heterogeneity," *IEEE Transactions on Emerging Topics in Computing*, 2018.

[16] Z. Zhu and P. Fan, "Machine learning based prediction and classification of computational jobs in cloud computing centers," *arXiv preprint arXiv:1903.03759*, 2019.
[17] H. Wang, J. Pannereselvam, L. Liu, Y. Lu, X. Zhai, and H. Ali, "Cloud workload analytics for real-time prediction of user request patterns," in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2018, pp. 1677–1684.
[18] N. Ding, C. Benoit, G. Foggia, Y. Bésanger, and F. Wurtz, "Neural network-based model design for short-term load forecast in distribution systems," *IEEE transactions on power systems*, vol. 31, no. 1, pp. 72–81, 2015.
[19] J. Panneerselvam, L. Liu, Y. Lu, and N. Antonopoulos, "An investigation into the impacts of task-level behavioural heterogeneity upon energy efficiency in cloud datacentres," *Future Generation Computer Systems*, vol. 83, pp. 239–249, 2018.
[20] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., White Paper*, pp. 1–14, 2011.
[21] I. S. Moreno, P. Garraghan, P. Townend, and J. Xu, "Analysis, modeling and simulation of workload patterns in a large-scale utility cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 208–221, 2014.
[22] J. Panneerselvam, L. Liu, N. Antonopoulos, and Y. Bo, "Workload analysis for the scope of user demand prediction model evaluations in cloud environments," in *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*. IEEE Computer Society, 2014, pp. 883–889.
[23] Y. Lu, J. Panneerselvam, L. Liu, and Y. Wu, "Rvlbpnn: A workload forecasting model for smart cloud computing," *Scientific Programming*, vol. 2016, 2016.
[24] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.
[25] X. Zhang, F. Shen, J. Zhao, and G. Yang, "Time series forecasting using gru neural network with multi-lag after decomposition," in *International Conference on Neural Information Processing*. Springer, 2017, pp. 523–532.
[26] X. Ouyang, P. Garraghan, D. McKee, P. Townend, and J. Xu, "Straggler detection in parallel computing systems through dynamic threshold calculation," in *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 2016, pp. 414–421.
[27] Google, "Google cluster data vi," *Google*, 2011, https://github.com/google/clusterdata/blob/master/ClusterData20112.md.
[28] W. Fang, Z. Lu, J. Wu, and Z. Cao, "Rpps: A novel resource prediction and provisioning scheme in cloud data center," in *2012 IEEE Ninth International Conference on Services Computing*. IEEE, 2012, pp. 609–616.

Yao Lu is currently working towards the PhD degree in computer science at University of Leicester. His current research is focused on energy efficient cloud systems and he has published his recent research works in journals and as book chapters. His research interests include Green Cloud Computing, Edge Computing, Artificial Intelligence, Big Data Analytics and 5G. He has won the best paper award in IEEE International Conference on Data Science and System, Exeter, 2018.



Lu Liu is the Head of School of Informatics and Professor of Informatics at the University of Leicester, UK. Prof. Liu received his Ph.D. degree from University of Surrey and M.S. degree from Brunel University. Prof. Liu's research interests are in the areas of data analytics, service computing, cloud computing, Artificial Intelligence and the Internet of Things. He is a Fellow of British Computer Society (BCS).

John Panneerselvam is a Lecturer in Computing at the University of Derby, United Kingdom. John received his PhD in computing from the University of Derby in 2018 and an MSc in advanced computer networks in 2013. He is an active member of IEEE and British Computer Society, and a HEA fellow. His research interests include cloud computing, fog computing, Internet of Things, big data analytics, opportunistic networking and P2P computing. He has won the best paper award in IEEE International Conference on Data Science and Systems, Exeter, 2018.

Bo Yuan is a Lecturer in Computing and a faculty member of Data Science Research Center (DSRC) with University of Derby, UK. He received the B.Sc. degree and the Ph.D. degree in computer science from Tongji University, China, in 2011 and 2017, respectively. He is an active member of IEEE and his research interests include Internet of Things, Online Social Networks, Mobile Computing, and Big Data Analytics.

Jiayan Gu received the BSc degree from University of Derby, UK, in 2018. She is currently working towards the PhD degree in the University of Derby, UK. Her research interests include service computing, edge computing and cloud computing. She is a Member of IEEE.

Nick Antonopoulos is currently the Vice Principal of Research and Innovation at Edinburgh Napier University. Nick holds a PhD in Computer Science from the University of Surrey in 2000. His research interests include Cloud Computing, P2P Computing, software agent architectures and security. Nick has over 18 years of academic experience and has published more than 150 articles in fully refereed journals and international conferences.