A Data-driven Approach to Identify Resource Bottlenecks for Multiple Service Interactions in Cloud Computing Environments

Lingxiao Xu[1], Minxian Xu*[2], Jian Wang[1], Richard Semmes[3], Qi Wang[4], Hong Mu[3], Shuangquan Gui[3], He Yu[4], Wenhong Tian*[1], and Rajkumar Buyya[5,1]

[1]School of Software and Information Engineering, University of Electronic Science and Technology of China, China
[2]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China
[3]Siemens Industry Software (Chengdu) Co., Ltd , China
[4]Siemens AG - Corporate Technology, Wuhan, China
[5]CLOUDS Lab, School of Computing and Information Systems, University of Melbourne, Australia

***Abstract*—Cloud service providers are providing resources including a variety of VM instances to support customers that migrate their services to the cloud. From the customers' perspective, selecting the appropriate amount of resources is tightly coupled with performance and cost. By identifying the potential resource bottlenecks in the early stage of the service deployment process, resource planning can be significantly optimized. To support system non-functional requirements in a better manner, we propose an approach to support the system's non-function requirements management concerning the multiple service interaction scenarios by identifying the potential resource bottleneck and optimizing the demanded resources. We also develop our proposed approach in the extended version of the CloudSim simulation toolkit. The proposed approach can predict the resource bottleneck for multiple service interactions, e.g. bottleneck in CPU or overloads in specific service, and provide guidance for resource planning. Evaluations with realistic data from Siemens MindSphere system and Alibaba Cloud show that our proposed approach can achieve good accuracy in terms of metrics, such as response time, queries per second (QPS) and resource usage.**

***Keywords*-Cloud; Service Interactions; Non-functional requirement; Resource bottleneck; MindSphere**

I. Introduction

The rapid development of cloud computing has made it be regarded as the fifth utility, like electricity, gas, and water [1]. Rather than assigning all tasks to a single local computer or a traditional computer cluster, cloud computing enables users to utilize computing or storage resources remotely, which provisions transparent and on-demand resources. In essence, the cloud is a networked computer paradigm based on virtualization techniques to improve resource usage. The pay-as-you-go model provided by cloud computing also helps the service providers and customers to start their business with minimal costs and eliminates the efforts to maintain the data centers. [2].

*Corresponding authors

Among all the benefits provided by cloud computing, some features are particularly attractive for customers. To be more specific, the first one is flexibility, which allows customers to acquire or release resources dynamically to fit their demands. The second is cost reduction, which means cloud computing service can convert capital expenditures to operational expenditures [3]. By utilizing the cloud, the expensive infrastructure like servers and professionals will not be the main concern of customers anymore. And the third advantage is the device and location independence [4], which supports customers to access computing resources anywhere and anytime, instead of using the specific interfaces to access the local server cluster and avoiding server maintenance time. Because of these benefits, many customers have started to migrate their business to the cloud, for instance, Alibaba has announced a statement that their services should be "All in Cloud". Currently, the prominent IT companies such as Amazon, Google, Microsoft, and Alibaba, have established their own cloud data centers and become cloud providers.

Though cloud computing has many advantages, based on the practice of utilizing resources of cloud computing, resource bottlenecks can still happen occasionally, like CPU, memory, and bandwidth [5] [6]. The reason is that when customers set up their services, they need estimate how much of the resources will be used for their services and reserve a specific amount of resources. However, due to the fluctuations of paperloads, overloads can happen thus lead to bottlenecks. When bottlenecks exist, the system performance, such as non-functional requirement (NFR) can be significantly influenced, for instance, CPU bottleneck can cause insufficient computing power and small number of parallel processes, memory bottleneck can lead to insufficient tasks or data waiting to be processed [7], and bandwidth bottleneck can trigger requests failed to be responded [8].

As bottlenecks can greatly affect the performance and user experience of cloud services [9], it is too important to identify the potential bottlenecks. However, different cloud services often provide different provisioned resource capacity, such as VM instances with different CPU, memory, and bandwidth, thus they may have bottlenecks under different load conditions. Besides, the bottlenecks of individual services are more difficult to be predicted based on different resource demands, running status and internal logic. If service providers can identify the potential bottleneck of each service and modify configurations, the costs and NFR can be satisfied.

However, identifying the bottlenecks at the early stage is not easy. Arranging testers to perform a large number of stress tests to evaluate the bottlenecks when changing resource configurations is not feasible. Apart from it, purchasing much more resources than required is not cost-effective and evaluation results may not be reproducible due to uncontrolled factors, such as network traffics.

To deal with the above challenges, using a simulation toolkit to simulate the real environment is a promising way. In this paper, we use CloudSim [10], which is a well-known cloud simulation toolkit that enables seamless modeling, simulation and experimentation of cloud computing [11]. With CloudSim, we can model the resource provisioning of various cloud infrastructure configurations and generate reproducible results. Large-scale simulations can also be easily conducted. Our motivation is to predict the potential resource and service bottlenecks based on a small-scale dataset of MindSphere [12][1], a cloud-based Internet-of-Things (IoT) open operating system from Siemens. We aim to generate various metrics to help MindSphere to plan hardware resources for their services. We also make efforts to make our solution as generic as possible, so it can be easily extended and migrated to other cloud platforms. To achieve these goals, we also have some challenges to address, including how to build a model for the realistic scenario of MindSphere, how to ensure that the system output results are consistent with the real test results, and how to make it a generic solution.

In this paper, we propose a data-driven framework to support the non-functional requirement, e.g. system performance, for multiple service interaction to identify the potential bottlenecks of service in the early stage. Based on CloudSim, we simulate the resource usage of interactive services, service interactions process. Our approach can predict the potential system bottleneck based on limited data collected from some real test cases and guide for the companies to optimize the resource configuration. Additionally, the system can help predict the response time and QPS when the system has reached the bottleneck.

Our key **contributions** are as follows:

- Proposing a framework to identify the potential bottleneck for multiple service interactions in the cloud to optimize resource planning for companies.
- Presenting an approach to model the resource utilization based on collected data, and guiding to predict the system behaviors under different loads.
- Extending CloudSim to model the service internal logic, Siemens MindSphere model and AliCloud.

[1]https://siemens.mindsphere.io/en

The rest of the paper is organized as: The related work is discussed in Section 2. We introduced our proposed framework, named, IRBS, in Section 3. Section 4 discusses the modeling of multiple service interaction scenarios, and the evaluation results based on MindSphere are demonstrated in Section 5. Finally, conclusions and future work are given.

II. Related work

To call cloud service more efficiently and reduce the cost, modeling services in the cloud, scheduling service in the cloud and optimizing resources in the cloud become pretty important. Neeraja et al. developed a data-driven system named PARIS, which can predict workload performance, resulting in costs and workloads across multiple cloud providers [13]. To decrease the task execution failure, Lattif et al. proposed DCLCA (dynamic clustering league championship algorithm) scheduling technique for fault tolerance awareness to address cloud task execution which would reflect on the currently available resources and reduce the untimely failure of autonomous tasks [14]. Sekaran et al. proposed a new meta-heuristic algorithm, named the dominant firefly algorithm, which optimizes load balancing of tasks among the multiple virtual machines in the Cloud server, thereby improving the response efficiency of Cloud servers that concomitantly enhances the accuracy of m-learning systems [15]. Cheng et al. proposed DRL-Cloud, a novel Deep Reinforcement Learning (DRL)-based RP and TS system, to minimize energy cost for large-scale CSPs with a very large number of servers that receive enormous numbers of user requests per day [16]. Nayak et al. used AHP (Analytic Hierarchy Process) as a decision-maker in the backfilling algorithm to choose the possible best lease from the given best-effort queue to schedule the deadline sensitive lease [17]. Priya et al. constructed a Fuzzy-based Multidimensional Resource Scheduling model to obtain resource scheduling efficiency in cloud infrastructure and increased utilization of Virtual Machines through effective and fair load balancing is then achieved by dynamically selecting a request from a class using Multi-dimensional Queuing Load Optimization algorithm [18]. However, these workers are not aiming at identifying the potential bottleneck of services in the system.

In order to find bottlenecks with low cost quickly, CloudSim is frequently used to build scenarios, set input and stimulate. For instance, Wickremasinghe wt al. developed CloudAnalyst, which is a CloudSim based tool for simulating large-scale Cloud applications to study the behavior of such applications under various deployment configurations [19]. Shi et al. used CloudSim to develop efficient energy-saving methods to reduce the huge energy consumption in the cloud datacenter [20]. Jung et al. proposed a simulation tool that supports the MapReduce model, implemented on CloudSim [21]. Belalem et al. proposed two approaches which aim at returning a better availability of Datacenters without deteriorating the performances for the answers of the users [22]. Alla et al. proposed Task Scheduling optimization using a novel approach

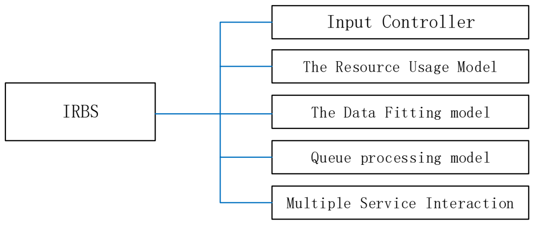


Fig. 1: Framework of IRBS

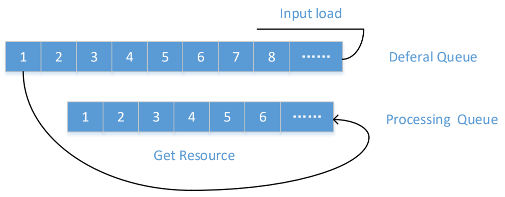


Fig. 2: Queue process

based on Dynamic dispatch Queues (TSDQ) and hybrid meta-heuristic algorithms, which is based on CloudSim and showed a great advantage in terms of waiting time, queue length, makespan, cost, resource utilization, degree of imbalance, and load balancing [23]. Sharma et al. proposed a modified particle swarm optimization (MPSO) task scheduling algorithm in order to optimize execution time, transmission time, makespan, transmission cost and load balancing of virtual machines and got the best cost as compared to original PSO on the CloudSim [24]. However, these workers do not model service internal logic. Besides, they do not provide the model for MindSphere and AliCloud.

III. IRBS Framework

To make our proposed approach to be reusable, correct and universal, we propose a framework to identify resource and service bottlenecks for multiple service interactions in Cloud, which we called IRBS (Identifying Resource Bottlenecks Solution). IRBS can evaluate different kinds of cloud services to achieve their availability and scalability. IRBS aims to predict the bottlenecks of the cloud service and identify the metrics when the service has reached the bottleneck. From the companies' perspective, selecting the appropriate amount of resources is tightly coupled with performance and cost, bottlenecks are the important reference metrics to improve the rationality of resource allocation. We provide a high-level summary below and then discuss the details in the subsequent sections.

We aim to make generic design, thus IRBS should be applicable to cloud services of different specifications by changing configurations. We also target to fit the data provided by Siemens to achieve good accuracy. Furthermore, when a new service is integrated into the system, our framework should also represent the behavior of the system.

The framework of IRBS has shown in Fig. 1, the framework contains five components: Input controller, Resource Usage Model, Data Fitting Model, Queue processing model, Multiple Service Interaction Modelling.

The *Input controller component* handles the input workloads. The workloads can be generated by this component, then, the concurrent loads enter into the system. The characteristic of loads is defined in the input properties, which will be loaded firstly. To make our framework to fit a realistic scenario, we consider the online scheduling policy, which we generate the loads then deal with the loads per unit time.

The *Resource Usage Model component* is the key component of IRBS. It produces the resource pool containing the CPU pool and Bandwidth pool, then simulates the real scenario to perform the loads' assignment. Based on this, results including real-time CPU usage, the real-time bandwidth usage, the initial response time can be achieved further. More details will be provided in Section IV.

The *Data Fitting model* component helps to calculate results. In the real scenario, there are various factors that can affect cloud service performance. For the hardware, the temperature, the voltage, and other elements can have huge impacts. As for the software, the scheduling policy and the adopted cloud service can also influence the NFR. IRBS uses Equation (1), which mainly calculates the response time based on the Resource Usage Model calculated. The response time is calculated as:

$$response\,time = finish\,time - submit\,time \tag{1}$$

*Queue processing mode*l controls two queues by monitoring the data in Resource Usage Model, which named deferral queue and processing queue. The deferral queue stores the load that is deferred when the resource pool is full so that the load can't be handled. The processing queue stores the loads under processing. The process is shown in Fig. 2. The component notifies the Input controller about the total loads in the system and then adjusts the load generator dynamically as feedback.

The *Multiple Service Interaction component* above will run in different services. Considered that if one service reaches the bottleneck, the other service will be affected, so IRBS designs the multiple service interaction to deal with the situation. By querying the queue processing model, the component will get the previous service queue and notify the next service, then achieve better effects.

IV. Multiple Service Interaction Modelling

A. Modelling Process

The operation process of the way to use this system is shown as the Fig. 3, the key steps are introduced as follows.

1) Collecting Data: The Data is provided by Siemens, which is from the MindSphere system and Alibaba cloud service. Certainly, IRBS is not limited in this scene, it is suitable for the service which comes down to the resource computation. The data mainly includes the load generation

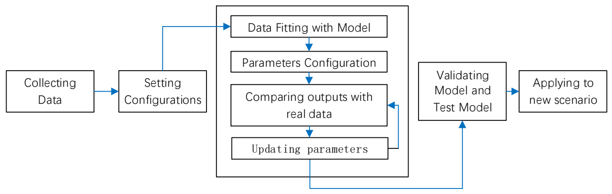


Fig. 3: Simulation Process

over time, CPU usage over time, bandwidth usage over time, response time over time and queries per second overtime. There are at least three sets of the data then the system will simulate real scenarios well.

2) Setting Configurations: IRBS doesn't just apply to one scenario, as a simulation system, it should be reusable. We adopt the properties as the tools to help the customers simplify input and implement differentiated input. We just need to modify the properties file and we will achieve different simulation results.

3) Data Fitting with Model: In the real scenario, taking CPU ability as an example. There is a lot of interference will affect CPU ability. To reduce the complexity of our modeling, we simplify the definition process. By abstracting the core number of CPU into parameter CPU-ability, IRBS distinguishes the computing ability of cloud services under different specifications. Consequently, we can use this parameter to measure the computing power in this service very well, as well as the bandwidth part.

4) Parameters Configuration: Because the equivalent CPU capacity of a core under different hardware specifications is different, we adjust the calculated resource mapping parameter by using the case that the compute resource is not fully loaded, the bandwidth part is also similar to this. after that the parameters CPU-ability and Bandwidth-ability in IRBS will be obtained.

5) Comparing outputs with real data: Certainly, using the default regression parameters in the properties will not get the correct answer. The hardware configuration of different cloud services are different, even cloud services of the same specification may vary slightly at different running phase, thus what we target is to fulfill the gap as much as possible with the right trends.

6) Updating parameters: We narrow this gap by adjusting the parameters in the linear regression model. Due to the situation of full load and non-full load, the simulation of the two sets of data alone will not achieve good performance. Therefore, a third set of full load conditions is needed to reflect the response of the component to the increasing load situation under the full load condition. After that, continuously adjust the parameters to ensure that the result pictures output by IRBS is almost the same as the pictures in the real test scene, the bottlenecks of the cloud service will be shown in the evaluations section.

7) Validating Model and Test Model: We can use another test dataset to validate our proposed model. Of course, the more data we have, the more precious the model will be. With limited data, the model can still guarantee accuracy. According to the initial output data, the user of the system can utilize this system to identify potential bottlenecks and predict some metrics of the cloud service.

8) Applying to new scenario: In our model, we try to make the models to be representatives of a series of similar service types, such as load generator services for generating loads, gateway services for forwarding loads, load balancing services for balancing loads, database related services for managing database queries. The basic scenario Siemens provides has five service, which will be introduced in section V. IRBS can be extended to the other scenario, whether it's changing the order or adding or removing services, or even applied to a brand new scenario only if it contains similar cloud service. We have tested its extendability and it is shown in section V.

B. Resource Usage Modelling

In the real scenario, both the CPU resources and bandwidth requirements of different workloads generated by different users are different, thus the companies aim to identify the bottlenecks of various cloud services. Based on this, we redefined the characteristics of the load generated by the users. Based on the data from Siemens, by using a random algorithm with normal distribution, the CPU resources and bandwidth resources required for the random generation of each load task are generated. Similarly, as mentioned in the previous sections, we abstracted the CPU resources and bandwidth resources in the real environment into specific values in the simulation system as a resource pool for processing loads. Due to the large magnitude of the load, The load characteristics is not easy to be fine-tuned. Therefore, We adjust the model by fine-tuning the multiple of the computing power corresponding to each core number.

The load stream generated by the input part firstly enters the delay queue. If resources are obtained, it enters the processing queue. When the load has completed, it is removed from the processing queue and record the time as the load finishing time. The load generation is controlled by reading the number of elements in these two queues.

We use an array to process the load and resource pool. The abscissa is used as the unit time length. Unit time is adjustable. The ordinate is used as resource usage. The characteristics of each load include the required CPU resources, the required bandwidth resources and the length of time, these resources are stored in the resource pool as small matrix blocks. Eventually, in this way, we can simulate the entire resource scheduling process.

In the Siemens scenario, or another scenario, there are multiple ECS (Elastic Compute Service) on the cloud server running MindSphere, and each ECS contains multiple pods, the basic unit for processing load is the pod. On this basis, we map ECS to the virtual machine, and pod to the container. The CPU resource pool uses the resources of the pod as a benchmark, and the bandwidth is based on the resources allocated by the entire service to perform resource scheduling. Owing to the

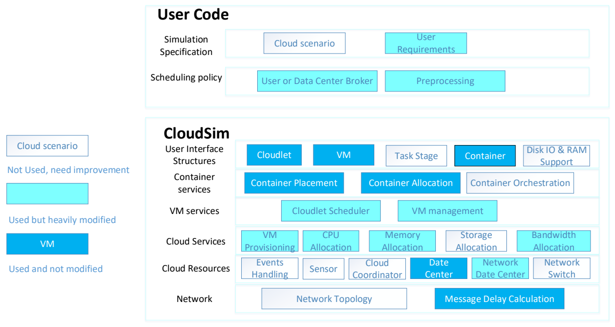


Fig. 4: Siemens Component

scenario there are multiple resource pools processing the same load, we use the round-robin algorithm to balance the loads.

When the resource pool is almost full, the pending loads obtained from the deferral queue cannot obtain resources, it also cannot be transferred from the deferral queue to the processing queue. In this case, the bottleneck of the cloud service under this configuration is reached, and all subsequent loads are delayed.

The current CPU utilization and bandwidth usage are obtained by reading the resource usage of each unit of time. The original average response time calculation method is shown in Equation (2), RT means response time, AVG means average response time. Generally, when resources are almost full, computing ability will decline due to competitive resources, so we used linear regression to calculate the basic average response time. Calculation Equation is shown in Equation (3) to get the final average response time, the parameters α and β are the regression parameters. According to the definition of QPS (Queries-per-second), Equation of QPS calculation is shown in Equation (4).

$$basic\,AVG = \frac{\sum_{load \in finished\,load} RT}{load\,number} \tag{2}$$

$$AVG = \alpha * basic\,AVG + \beta \tag{3}$$

$$QPS = \frac{total\,average\,Response\,time}{load\,Number\,Unit\,Time} \tag{4}$$

C. Extension of CloudSim for Siemens scenarios

By following CloudSim approach, we have implemented the definition of basic classes by inheriting cloudlet, container, containerVM and host. We use the queueing method, and refer to the resource scheduling method to design our scheduling policy. All of these are shown in Fig. 4.

Considering the scenario provided by Siemens, there is a mechanism similar to feedback which the CloudSim does not have. The scheduling logic of CloudSim is offline, it submits all the cloudlets firstly, then deals with them later. The policy is not able to the feedback mechanism. At the same time, submitting all tasks at one time will also bring a lot of unfavorable factors, such as high computer memory requirements and difficulty to control time accuracy. Similarly, CloudSim has virtual machine migration and container migration in resource scheduling, these are not suitable for our scenario. We have made improvements and simplifications on the basis of these, making scheduling more suitable for this scenario, and finally implementing our system.

TABLE I: Gateway Tests

Case	Load	Pod CPU utilization	Throughput	Average response Time	Bandwidth
1	1500 vu	38.5 cores	10000 calls/s	50ms	300 Mbps
2	2000 vu	46 cores	11300 calls/s	75ms	342 Mbps
3	3000 vu	47 cores	11600 calls/s	158ms	352 Mbps

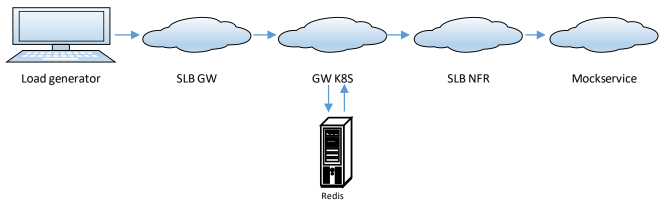


Fig. 5: Simulation Process

V. Performance Evaluation

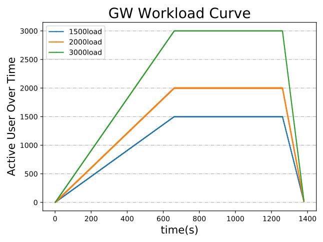
Siemens has deployed their services on Alibaba cloud to realize the operation of the entire business by using its MindSphere system. The Siemens scenario is shown in Fig. 5. The load generator is the service that generates the loads for the stem. SLB GW is used as the Siemens MindSphere task scheduler, which is responsible for maintaining the load balancing and performing task distribution. The VMs and containers are responsible for processing the loads among different services, and performing database storage and reading operations on Redis. SLB NFR is used to monitor various metrics in the entire system, and Mockservice BlackBox component that generates as a fixed processing delay component, e.g. 300 ms. Gw k8s is the core service that connects the outside and inside of the system.

In our simulations, each unit time represents one second by default, and the time accuracy can be adjusted to make it more fine-grained. However, if the accuracy is improved, the simulation takes longer running time and consumes more memory resources. To balance the trade-offs between accuracy and running costs, we simulate and collect the CPU usage, bandwidth usage every unit time. And then, we draw the result based on the obtained metrics values. After a series of tuning and parameters updating, the variance of simulated results and real data are reduced to be small enough. The simulation is completed for the situation under this configuration.

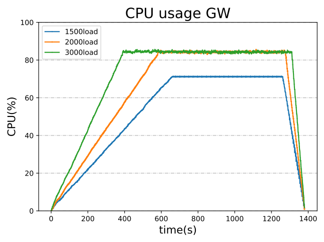
Based on the results, we can roughly know when the bottleneck will exist under the specific loads and then adjust the program configuration file according to the corresponding load size where the bottleneck occurs. In this way, the number of the loads when CPU resources reach the bottleneck or

TABLE II: Hardware Configuration

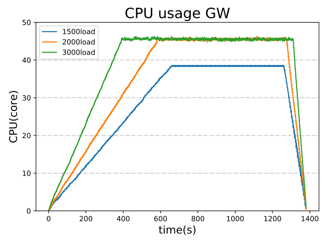
Case	Load	ECS number	Pod number	ECS cores	Pod cores	Bandwidth	Other service response time
1	1500 vu	9	18	8 cores	3 cores	350 Mbps	about 100 ms
2	2000 vu	9	18	8 cores	3 cores	350 Mbps	about 100 ms
3	3000 vu	9	18	8 cores	3 cores	350 Mbps	about 100 ms
4	3020 vu	20	40	8 cores	3 cores	650 Mbps	about 100 ms
5	4000 vu	16	32	8 cores	3 cores	800 Mbps	about 100 ms
6	4000 vu	20	40	8 cores	3 cores	610 Mbps	about 100 ms



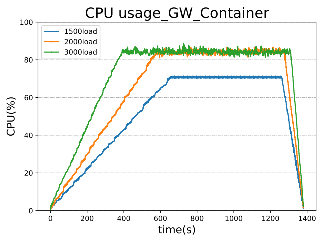
(a) GW Workload Curve



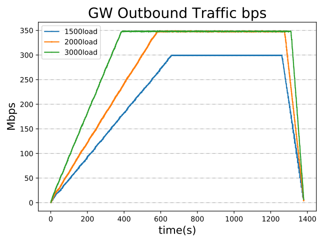
(b) CPU usage



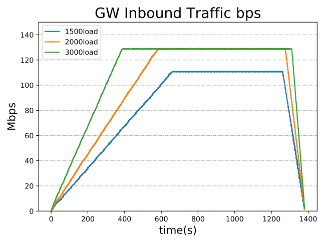
(c) CPU usage



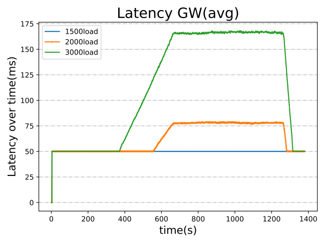
(d) Container CPU usage



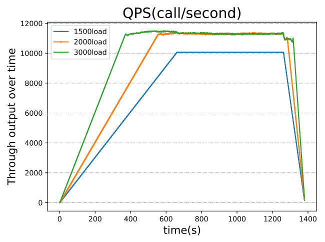
(e) outbound traffic



(f) inbound traffic



(g) Latency GW



(h) QPS

Fig. 6: The metrics of different loads under Siemens specification

the Bandwidth reach the bottleneck can be obtained in this configuration. Additionally, we can use IRBS to predict some scenarios without evaluations under real tests, for example in what CPU configuration or bandwidth configuration the QPS will reach 20000, and optimize resource planning without changing hardware specifications, only change the cores or the max bandwidth.

In the scenario provided by Siemens, GW K8S is the core component to be evaluated, as the whole data is confidential, only the key data is shown in Table I. In this scenario, there are 9 ECS and 18 pods in total for the resource provisioning. Each ECS is equipped with 8 cores, each pod has 3 cores, and the bandwidth is 350 Mbps. The detail configuration is shown in Table II. In the same hardware configuration, there are two cases in general, which are non-full load and full load.

A. Non-full load

The case is when the cloud service mostly runs in the real scenario, both the CPU and the inbound and outbound traffic have not reached the bottleneck, and all loads can respond in time to the cloud service.

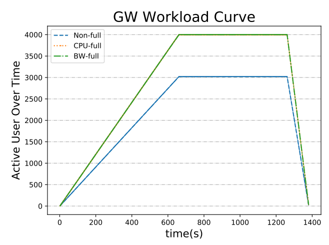
In Fig. 6 we give a specific example to illustrate. The input loads follow this trend. After 11 minutes, the loads reach the peak point, then the input controller component keeps the peak load for another 10 minutes, and then the loads gradually decrease in the last 2 minutes.

Under the above non-full load situation, the comparison between the chart in the test report provided by Siemens and the test chart obtained through system simulation is shown in Fig.6. When the load reaches 1500 users, the CPU utilization is about 70%, the number of corresponding cores usage is about 38.5 cores, and the outbound traffic (Upstream and downstream directions are opposite) is about 300 Mbps. Since Siemens did not provide a specific image about the response time but provided a reference value of 50ms, this value is also well reflected. QPS also showed a rising straight line, and reached the peak when the load reached the peak.

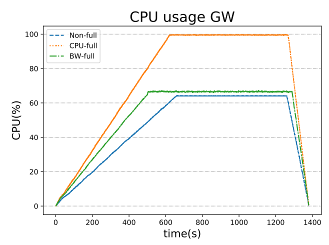
B. Full load

Considering the fully loaded bandwidth is fully loaded as another example. The case is shown in Fig. 6, which is as same as the case of non-full load, except that the number of loads at the peak becomes 2,000 and 3,000. In this case, we can observe that the CPU utilization has not reached the full load, but the bandwidth rate has reached the full load earlier and reached 350 Mbps, which means bandwidth can be the bottleneck rather than CPU in this case.

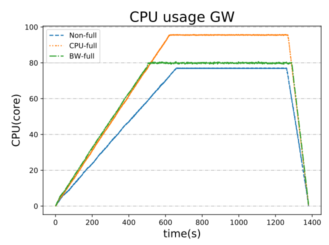
Observing the evaluations, we can also find that when the outbound traffic reaches the bottleneck and the saturation happens, the loads which arrive latter can't enter the service.



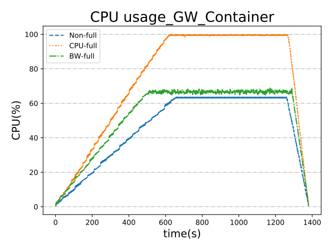
(a) GW Workload Curve



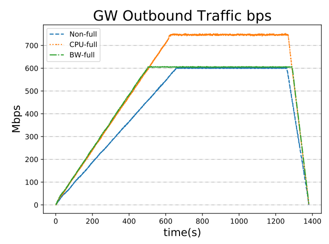
(b) CPU usage



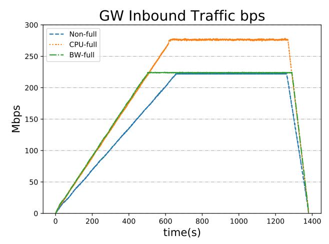
(c) CPU usage



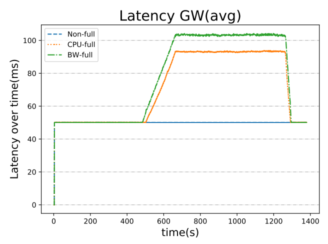
(d) Container CPU usage



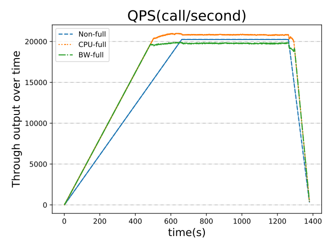
(e) outbound traffic



(f) inbound traffic



(g) Latency GW



(h) QPS

Fig. 7: The metrics of different loads when QPS is 20000 calls/s.

Although CPU is not fully loaded, the CPU utilization also reaches the peak which becomes about 90% and the QPS also reaches 11600 calls/s.

The results obtained by the simulation system is nearly the same as the real test results. Considering that resource competition will occur when resource usage approaches the bottleneck value, resulting in performance degradation, the average response time tends to rise in advance. Meanwhile, it is observed that when the bottleneck point is reached, the QPS remains at its peak, but the load is still rising at this time, so the increase in response time at this time must be consistent with the load. The linearity of the response time is restricted by this restriction. So we can calculate regression parameters. Finally, a relatively good simulation effect was obtained. According to the results, it can be observed that when the load is close to 1,800, it is the bottleneck in this configuration.

C. Bottleneck Prediction

In the above cases, they have demonstrated how to identify the bottleneck in specific configurations. IRBS can predict the satisfaction of various metrics based on the parameters of the current specifications, that is, only the core and number of ECS and pods and the bandwidth are changed. Taking QPS as an example, in Fig. 7 shows various situations when the QPS reaches 20,000, and the configuration of each situation is shown in TableII.

The first case is that the CPU and bandwidth are not fully loaded. In this case, we can notice that when the number of loads reaches 3020, the QPS is close to 20,000 calls/s. At this time, the peak response time of GW service is 50ms, and other services are also at In the case of not fully loaded, the total response time of the other parts is close to 100ms, so the total response time is about 150ms. According to Equation (4) calculation, we can conclude that the prediction result is accurate.

The second and third cases are when the CPU or bandwidth reaches full load. In these cases, the CPU or bandwidth utilization is about 100%. Under this situation, the peak response time of the GW service is about 100ms. And the total response time is about 200ms, which shows that the results are also accurate. Then the companies can use these results to optimize their infrastructure configuration.

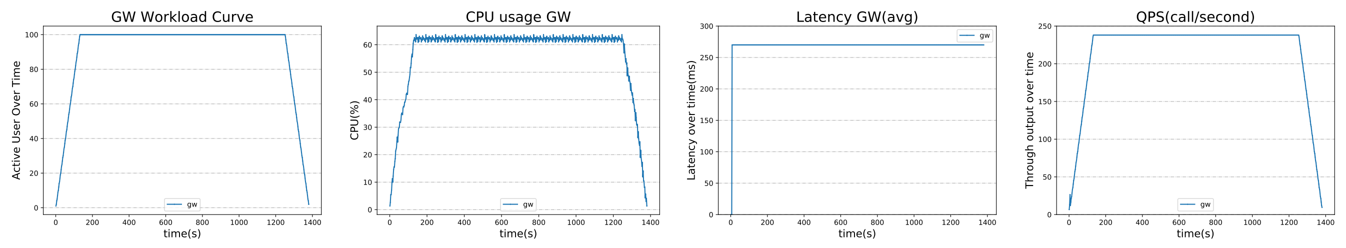
D. Scalability

Finally, we want to show that this system is not only suitable for the discussed scenario. In another scenario proposed by Siemens, there are only three services, namely load generation, GW, and mockservice. Due to changes in the hardware resources for support services, we can re-adjust the parameters, select different parts by adjusting the VM and container capacity, and control the execution order by adjusting the id. The results are shown in Fig. 8.

The difference from the above is that only the properties configuration file is changed, which changes the number, type, and order of cloud services. By comparing data with a real scenario, the data from our simulated results can fit the data in the table well.

VI. Conclusions and Future Work

In this paper, we propose a data-driven approach called IRBS, which can predict resource and service bottlenecks in multiple service interaction scenarios. We also developed a generic modeling process to ensure good accuracy while having good reusability and scalability. Based on the dataset provided by Siemens MindSphere system, our simulated results show that our approach can effectively identify the potential bottlenecks and achieve good accuracy for adopted metrics. The dominant advantage of the proposed approach is



(a) GW Workload Curve (b) CPU usage (c) Latency GW (d) QPS

Fig. 8: Extended scenario

easy to operate and provides a good basis for the customers' ability to configure hardware resources.

As future work, we would like to reduce the consumption of computer resources as much as possible under the condition of improving the accuracy of simulation time. It is intended that each resource pool uses two arrays to alternately or other better way process the load, and the purpose of optimizing the scheduling process is achieved by deleting the resources occupied by the expired processed load. We also would like to investigate other datasets or a broad range of conditions to evaluate our proposed approach.

Acknowledgment

This research is partially supported by National Natural Science Foundation of China, with Grant ID 61672136 and 61828202. The authors would like to thank teams in Siemens Industry Software Co., Ltd, China, for their discussion and comments on this work.

References

[1] M. Xu and R. Buyya, "Brownout approach for adaptive management of resources and applications in cloud computing systems: a taxonomy and future directions," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–27, 2019.

[2] M. N. Birje, P. S. Challagidad, R. Goudar, and M. T. Tapale, "Cloud computing review: concepts, technology, challenges and security," *International Journal of Cloud Computing*, vol. 6, no. 1, pp. 32–57, 2017.

[3] K. Subramanian, "Recession is good for cloud computing microsoft agrees," 2009.

[4] D. Farber, "The new geek chic: Data centers," *CNET News*, vol. 25, 2008.

[5] R. Buyya, S. N. Srirama, G. Casale, R. Calheiros, Y. Simmhan, B. Varghese, E. Gelenbe, B. Javadi, L. M. Vaquero, M. A. Netto *et al.*, "A manifesto for future generation cloud computing: Research directions for the next decade," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–38, 2018.

[6] M. Xu, W. Tian, and R. Buyya, "A survey on load balancing algorithms for virtual machines placement in cloud computing," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 12, p. e4123, 2017.

[7] A. Yazdanbakhsh, B. Thwaites, H. Esmaeilzadeh, G. Pekhimenko, O. Mutlu, and T. C. Mowry, "Mitigating the memory bottleneck with approximate load value prediction," *IEEE Design & Test*, vol. 33, no. 1, pp. 32–42, 2016.

[8] K. Lai and M. Baker, "Nettimer: A tool for measuring bottleneck link bandwidth." in *USITS*, vol. 1, 2001, pp. 11–11.

[9] O. Ibidunmoye, F. Hernández-Rodriguez, and E. Elmroth, "Performance anomaly detection and bottleneck identification," *ACM Computing Surveys (CSUR)*, vol. 48, no. 1, pp. 1–35, 2015.

[10] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[11] R. Malhotra and P. Jain, "Study and comparison of cloudsim simulators in the cloud computing," *The SIJ Transactions on Computer Science Engineering & its Applications*, 2013.

[12] D. Petrik and G. Herzwurm, "iiot ecosystem development through boundary resources: a siemens mindsphere case study," in *Proceedings of the 2nd ACM SIGSOFT International Workshop on Software-Intensive Business: Start-ups, Platforms, and Ecosystems*, 2019, pp. 1–6.

[13] N. J. Yadwadkar, B. Hariharan, J. E. Gonzalez, B. Smith, and R. H. Katz, "Selecting the best vm across multiple public clouds: A data-driven performance modeling approach," in *Proceedings of the 2017 Symposium on Cloud Computing*, 2017, pp. 452–465.

[14] M. S. A. Latiff, S. H. H. Madni, M. Abdullahi *et al.*, "Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm," *Neural Computing and Applications*, vol. 29, no. 1, pp. 279–293, 2018.

[15] K. Sekaran, M. S. Khan, R. Patan, A. H. Gandomi, P. V. Krishna, and S. Kallam, "Improving the response time of m-learning and cloud computing environments using a dominant firefly approach," *IEEE Access*, vol. 7, pp. 30 203–30 212, 2019.

[16] M. Cheng, J. Li, and S. Nazarian, "Drl-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers," in *in Proceedings of the 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2018, pp. 129–134.

[17] S. C. Nayak and C. Tripathy, "Deadline sensitive lease scheduling in cloud computing environment using ahp," *Journal of King Saud University-Computer and Information Sciences*, vol. 30, no. 2, pp. 152–163, 2018.

[18] V. Priya, C. S. Kumar, and R. Kannan, "Resource scheduling algorithm with load balancing for cloud service provisioning," *Applied Soft Computing*, vol. 76, pp. 416–424, 2019.

[19] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *Proceedings of the 2010 24th IEEE international conference on advanced information networking and applications*. IEEE, 2010, pp. 446–452.

[20] Y. Shi, X. Jiang, and K. Ye, "An energy-efficient scheme for cloud resource provisioning based on cloudsim," in *Proceedings of the 2011 IEEE International Conference on Cluster Computing*. IEEE, 2011, pp. 595–599.

[21] J. Jung and H. Kim, "Mr-cloudsim: Designing and implementing mapreduce computing model on cloudsim," in *Proceedings of the 2012 International Conference on ICT Convergence (ICTC)*. IEEE, 2012, pp. 504–509.

[22] G. Belalem, F. Z. Tayeb, and W. Zaoui, "Approaches to improve the resources management in the simulator cloudsim," in *International Conference on Information Computing and Applications*. Springer, 2010, pp. 189–196.

[23] H. B. Alla, S. B. Alla, A. Touhafi, and A. Ezzati, "A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment," *Cluster Computing*, vol. 21, no. 4, pp. 1797–1820, 2018.

[24] S. K. Sharma and N. Kumar, "A modified particle swarm optimization for task scheduling in cloud computing," *Available at SSRN 3349598*, 2019.