Wu et al. Journal of Cloud Computing: Advances, Systems and Applications (2021) 10:30
https://doi.org/10.1186/s13677-021-00242-w

Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

Intent-driven cloud resource design framework to meet cloud performance requirements and its application to a cloud-sensor system

Chao Wu*, Shingo Horiuchi, Kenji Murase, Hiroaki Kikushima and Kenichi Tayama

Abstract

In cloud service delivery, the cloud user is concerned about "what" function and performance the cloud service could provide, while the cloud provider is concerned about "how" to design proper underlying cloud resources to meet the cloud user's requirements. We take the cloud user's requirement as intent and aim to translate the intent autonomously into cloud resource decisions. In recent years, intent-driven management has been a widely spread management concept which aims to close the gap between the operator's high-level requirements and the underlying infrastructure configuration complexity. Intent-driven management has drawn attention from telecommunication industries, standards organizations, the open source software community and academic research. There are various application of intent-driven management which are being studied and implemented, including intent-driven Software Defined Network (SDN), intent-driven wireless network configuration, etc. However, application of intent-driven management into the cloud domain, especially the translation of cloud performance-related intent into the amount of cloud resource, has not been addressed by existing studies. In this work, we have proposed an Intent-based Cloud Service Management (ICSM) framework, and focused on realizing the RDF (Resource Design Function) to translate cloud performance-related intent into concrete cloud computation resource amount settings that are able to meet the intended performance requirement. Furthermore, we have also proposed an intent breach prevention mechanism, P-mode, which is essential for commercial cloud service delivery. We have validated the proposals in a sensor-cloud system, designed to meet the user's intent to process a large quantity of images collected by the sensors in a restricted time interval. The validation results show that the framework achieved 88.93 ~ 90.63% precision for performance inference, and exceeds the conventional resource approach in the aspects of human cost, time cost and design results. Furthermore, the intent breach prevention mechanism P-mode significantly reduced the breach risk, at the same time keeping a high level of precision.

Keywords: Intent-driven cloud management, Cloud resource, Cloud performance, Intent breach prevention

Introduction

According to our interviews with cloud service operators and analysis of the form of current cloud service delivery, a gap has been existing between the cloud user and the cloud provider, such that the cloud user and cloud provider "speak different languages" during the cloud service delivery. On the one hand, the cloud user is concerned about the "what", that is the functionality, performance etc. that the cloud service is able to provide, and the cloud user tends to communicate with the cloud provider in terms of these "service-level" requirements e.g. the cloud-based web service needs to be able to "process 1000 requests per second,". On the other hand, the cloud provider needs to know what types of VM are needed to realize the functionality required, and how much cloud computation resource, e.g. number of

* Correspondence: chao.wu.ex@hco.ntt.co.jp
Tokyo, Japan

vCPUs, is necessary to meet the performance requirements, and thus the cloud provider tends to communicate in this "resource-level language". In this work, we focus on solving the complexity required to translate the user's performance requirements into the amount of resources needed.

To close the gap between the cloud user and the cloud provider, translation from the cloud user's service-level requirements to the resource requirements is necessary. Currently, the "translation" relies heavily on human experience and the skills either of the cloud user or of the cloud provider. As a result, if relying on the user to translate their service-level requirement to resource demands, it will result in a skills barrier for the user which prevents users with inadequate experience and skills of cloud infrastructure from using the service. Conversely, if relying on the cloud provider to translate the service requirements into the resource demand, the human-based translation leads to high human cost, long service lead time and potentially low resource efficiency for the cloud provider in delivering the cloud service.

In this work, we take the user's service-level requirement as the intent of the cloud user, and we aim to resolve the complexity of translating the cloud user's performance-related intent, into concrete cloud resource requirements, e.g. the number of vCPUs, and the size of vMemory that satisfies the intent. The main contributions of our work are as follows:

(1) We have investigated and analyzed the current activities concerning intent-based management from the standardization organizations and open source software communities along with academic studies, and summarized their definitions, scope, and application scenarios of intent-based management. From these investigations we can see that intent-based management to meet the user's performance requirements has barely been studied.
(2) We have proposed an Intent-based Cloud Management (ICSM) framework and focus on the realization of the RDF (Resource Design Framework), which is able to translate the user's intent about performance into cloud computation amount, e.g. the amount of vCPU, memory, etc., required. The validation experiment shows that the RDF is able to realize a precision of 88.9 ~ 90.6% for resource design, and reduce time costand human cost compared to the conventional resource design approach.
(3) In commercial service delivery, breach of intent will result in a serious penalty to the cloud provider as well as a decrease in user satisfaction, so we have proposed an intent breach prevention mechanism, P-mode for RDF, which lower the intent breach risk. The validation experiment shows that P-mode reduces intent breach risk in the aspect of breach extent, breach duration and breach time by 42.25%, 66.05% and 43.02% respectively.
(4) We have conducted an experiment in a cloud-sensor system scenario to validate RDF's precision, reduction of human and time cost, and resource design results compared to the conventional resource design approach. We have also validated the intent breach prevention effect of the proposed breach prevention mechanism for RDF.

The composition of this work is as follows. In section II, we introduce the background and the challenges of cloud service management, and the demand for an intent-driven cloud management framework. In section III, we introduce the related intent-driven management studies from the perspectives of academia, standardization bodies and the open source software community. In section IV, we introduce our proposed ICSM (Intent-based Cloud Service Management) framework, with a focus on the RDF (Resource Designer Function), which translates the intent into cloud computation resource amount. We also introduce the proposal of an intent breach prevention mechanism for RDF in this section. In section V, we describe our validation experiment of RDF in a cloud sensor system scenario. In section VI, we give the validation results of RDF. In section VII, we conclude the work and introduce plans for future work.

Background and problem statement

An increasing number of individual users, enterprises and organizations are utilizing a cloud for conducting computing tasks, saving and sharing content, hosting end user-facing services, etc. Although cloud service business models vary hugely depending on different scenarios, we can conclude that there are usually 2 main parties in a cloud service business model as following.

Cloud provider: the party who is responsible for providing the cloud resources.

Cloud user: the party who consumes the cloud resources (the virtual machine (VM), virtual central processing unit (vCPU), virtual memory, etc.) to process their computation workload.

The cloud user and the cloud provider can be from the same organization but play different roles, e.g., an application engineer who is the cloud user and a cloud engineer who is the cloud provider in the same company, or in different organizations, e.g., an engineer of an e-commerce company who is the cloud user and the public cloud service provider who is the cloud provider.

Between the 2 parties, a profound gap has been existing, in that *cloud users speak service-layer language while cloud resource providers speak resource-layer language.* As shown in Fig. 1, when requesting cloud services,

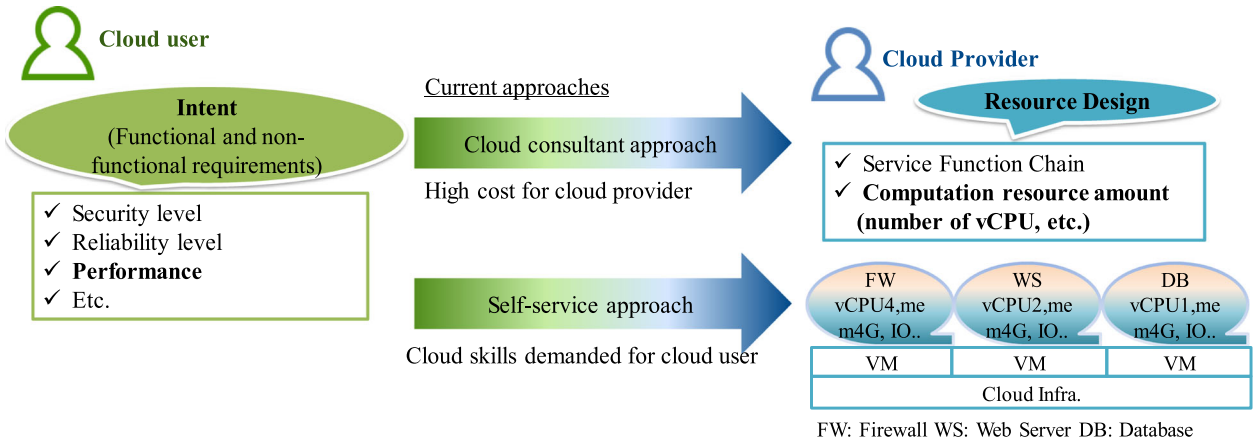

Fig. 1 Cloud user and cloud provider speaking different layers of "language"

cloud users tend to tell the cloud provider about their service-layer functional and non-functional requirements, such as a service's functionality requirements, requirements about ability to handle workload, security requirements, and reliability requirements. For example, an engineer from an e-commerce company will be concerned about whether the e-commerce service that runs on the cloud is able to handle 1000 transactions per second, and provide the highest levels of security and reliability, rather than how these requirements are met and how the cloud resource that accommodates the e-commerce service is implemented. In other words, the cloud user is more concerned about "what" requirements need to be met when using the cloud rather than "how" to meet the requirements. We call the cloud user's service layer requirements *intent* in this work. In contrast, cloud providers, when providing the cloud services to the cloud user, need to decide concrete cloud resource configurations, such as the necessary types of instance, the SFC (Service Function Chain), the amount of vCPU, and memory to be allocated to each instance in the SFC to meet the cloud user's service-layer requirements. In other words, the cloud provider needs to translate the user's intent about the cloud service ("what") into "how" the service is implemented to meet the user's requirements. In Table 1, we give 3 practical examples that illustrate the gap between the cloud users' intents and the cloud provider's concerns. Specially, in this work, we put our focus on how to translate the user's requirements (intent) about performance (the ability to handle workload) to the amount of resource (number of vCPU, size of vMemory) that meets these requirements.

According to our investigation into current cloud service delivery, there are two main approaches to translate the user's intent about the cloud service ("what") into "how" the service is implemented: the *cloud-consultant approach* and *self-service approach*. In the *cloud-consultant approach*, cloud users are supported by cloud consultants from the cloud provider, who collect users' service-layer requirements and determine the resource details accordingly. However, this puts a huge workload on cloud operators, causes high OPEX and needs a relatively long time to deliver service. In the *self-service approach*, cloud users are provided with a management interface to manage cloud resources and need to decide their required resource by themselves on the basis of their service requirements. Thus, users need to be skilled in cloud issues in order to decide and manage cloud resources. In both approaches, translation of service layer requirements into resources heavily relies on human decisions.

The translation of performance-related intent into amount of cloud resource involves understanding both the cloud-based applications' workload patterns, and the underlying cloud resource configuration patterns, and most importantly, the complex relationship between the two. What is more, the increasing variety of service requirements and high flexibility in cloud resource configuration add to the complexity of the decision-making process. Thus, resource design support for cloud resource decisions is essential for lowering the barriers for the cloud users to use clouds, as well as for reducing the service delivery lead time and cost of cloud resource design for cloud providers.

Related work

The concept of intent-driven management has drawn attention and been applied in various network management areas, such as SDN, wireless networks etc. It aims to free the network operator from complex underlying infrastructure configuration issues by letting the operator define "what" state the managed entity is intended to be in, rather than "how" to achieve the state. In this

Table 1 Examples where cloud user concerns are about "What" while the cloud provider needs to decide "how"

Cloud type	Scenario	Cloud user	Cloud provider	Cloud user's concerns (intents)	Cloud provider's concerns (resource decisions)
Public cloud	MLaaS	Common MLaaS users	MLaaS provider	Time taken to accomplish a given training task, etc.	Amount of computation resources to be allocated in order to meet the training time restrictions from the user
NFV implemented on Private cloud	vEPC	Network traffic management department in a CSP	Virtualization platform management department in a CSP	- High reliability of the communication service - Packet processing capacity and latency	- SFC composition (e.g. add standby SFC) to realize the high reliability requirement - Amount of computation resources to be allocated in order to meet the packet processing capacity and latency requirement
Public cloud	Ecommerce service on cloud	E-commerce company	Public cloud provider	- High reliability, security levels of the communication service - Packet processing capacity and latency	- SFC composition (e.g. add standby SFC, add packet filtering instance to SFC) to realize the high reliability and security requirements - Amount of computation resources to be allocated in order to meet the packet processing capacity and latency requirement

section, we will introduce current work of intent-driven management from the perspectives of academia, standardization organization activities and open source software community activities.

Latest studies about intent-driven management in standard organizations

ETSI ENI defines in its work item ENI-005 [1] that intent uses a controlled language to express the goals of the policy, without describing how to accomplish the goals. The intent could be applied in various network domains, such as wireless network, data center, etc. It has also started a work item "Intent Aware Network Autonomicity" [2] in the year 2019 in which the intent translation process, intent management lifecycle, external interfaces of intent translation function are studied.

TMForum has established the "Autonomous Network Whitepaper Release 2" [3] in the year 2020, which aims to give guidelines to realize closed-loop management in 4 domains including resource domain closed-loop management, service domain closed-loop management, business domain closed-loop management and user-domain closed-loop management. Intent-driven management is defined as one of the key technical mechanism to realize the goal.

3GPP SA5 in its release17 has studied the concepts, scenarios, and solutions for intent-driven management, which enables simplifying the management interfaces (TR28.821) [4], and the normative work for intent-driven management has started in TS28.312 [5]. They define intent as "a desire to reach a certain state for a specific service or network management workflow with more focus on "what" than traditional management". 3GPP has defined a wide variety of intent-driven management scenarios based on the following categorization:

Intent categorizes based on user types:

- Intent from Communication Service Customer
- Intent from Communication Service Provider
- Intent from Network Operator

Intent categorizes based on management scenario types

- Intent for network and service design/planning
- Intent for network and service deployment
- Intent for network and service maintenance
- Intent for network and service optimization/assurance

ETSI ZSM has named intent-driven management as one of the closed-loop goal in the work item "Closed-loop automation";

"Enablers" [6]. Detailed studies about the intent and its role in closed-loop automation are still in progress.

Open source communities

The most influential open source intent-driven management implementations are from Open Network Operating System (ONOS) and OpenDayLight.

ONOS defines intent as an immutable model object that describes an application's request to the ONOS core to alter the network's behavior, and it has a module called Intent Framework [7]which handles the intent by compiling intent into ONOS configurations and installing the intent.

OpenDayLight [8] has introduced a Network Intent Composition (NIC), which is an interface that allows clients to express a desired state in an implementation-neutral form. They have implemented the NIC through which the operator is able to manipulate the network topology, QoS in a vendor-neutral approach.

Both of the open source software implementations are mainly targeted at SDN management area.

Academic activities

Intent-driven management has also been studied in academia. Numerous studies have been done in different application scenarios of intent-driven management. The main scenarios include: (1) intent-driven SDN management, which is highly related to the work of ONOS, Opendaylight, for example [9, 10]; (2) Natural language-based intent translation e.g. [11]; (3) Traffic control, e.g. [12].

In this work, we mainly introduce related work especially focused in intent-driven management in cloud area.

In [13], the author has proposed an intent-based approach to defining the north-bound interface between the VIM (Virtual Infrastructure Manager) and the higher management and orchestration layers defined by the ETSI MANO framework. The authors aim to define vendor-independent, intent-based interfaces so that making the service requests agnostic to the specific technology adopted by each SDN domain.

In [14], the authors have proposed an intent-based network solution that refines intents into service chains of VNFs by employing soft goal interdependency graphs and clustering. The paper has studied a scenario where a security-related intent is interpreted in to SFC using the proposed technique.

In [15], the author has proposed an over-the-top intent-based networking framework that aims to automate the 5G slice resource provisioning. The natural language based intent is mapped to concreate resources according to pre-defined SFC template.

In [16], the author has mainly discussed how to compose an intent using logical labels and proposed a label management system to map the intent into an entity group. The entity group in this work refers to the relationship between logical or physical resource.

In the most recent years (2020~), there have come more academic studies about intent-based management, from which we have selected the four most relative works in cloud management area.

In [17], the authors have addressed the challenge that managing diverse security equipment requires significant security knowledge of the various network security equipment for the security service provider/consumer. They have proposed an IBCS (Intent-Based Cloud Services for Security Applications) that translates intents of security service consumers into low-level security policies and maps the required security functionality into concrete virtual machines in the cloud environment.

In [18], the authors have focused on the task to match between cloud resources and processing tasks in accordance with multiple attributes as well as treats the interests of both provider and the user fairly. They have proposed an intent-based resource matching strategy in cloud that is aimed to improve the accuracy and efficiency of matching and overall satisfaction of both parties.

In [19], the authors have addressed the complexity for the network service provider to manually organize the systems to fulfill the service requirements, dependencies, and constraints derived from the system components. They have proposed a search-based system that translates the customer's service requirement which is a high-level, abstract topology with quantitative constraints into concrete network components and network connections.

In [20], the authors have addressed the challenge that IoT users who are not experts in IoT-cloud environments may find it difficult to set up the right configuration for diverse IoT equipment which supports different protocols and services. They have proposed an IoT device configuration translator for intent-based IoT-cloud services to solve the problem. The translator translates the IoT user's intent natural language on IoT device configuration into low-level IoT configurations in XML format of YANG (Yet Another Next Generation).

We can see from the investigation of related work that although intent-based management has been studied by standards organizations, open source communities and academic bodies for various areas including cloud management, most of the focus of intent-based management's application in the cloud management area has been the translation from a non-performance related intent into an SFC template. To the best our knowledge, translation of cloud performance-related intent into cloud resource amount has barely been studied. However, as introduced in sections I and II, translating performance-related intent is a highly complex task that demands operator's knowledge and experience of the relationship between performance, various cloud-based workloads and numerous configuration patterns of resource amounts, and we will address the challenge by proposing the RDF to assist in the translation of performance-related intent.

ICSM framework and RDF

To solve the challenge in Section II, we have proposed an Intent-based Cloud Management (ICSM) framework (Fig. 2) to automate the translation process from intent to resource requirements in our preliminary work [21].

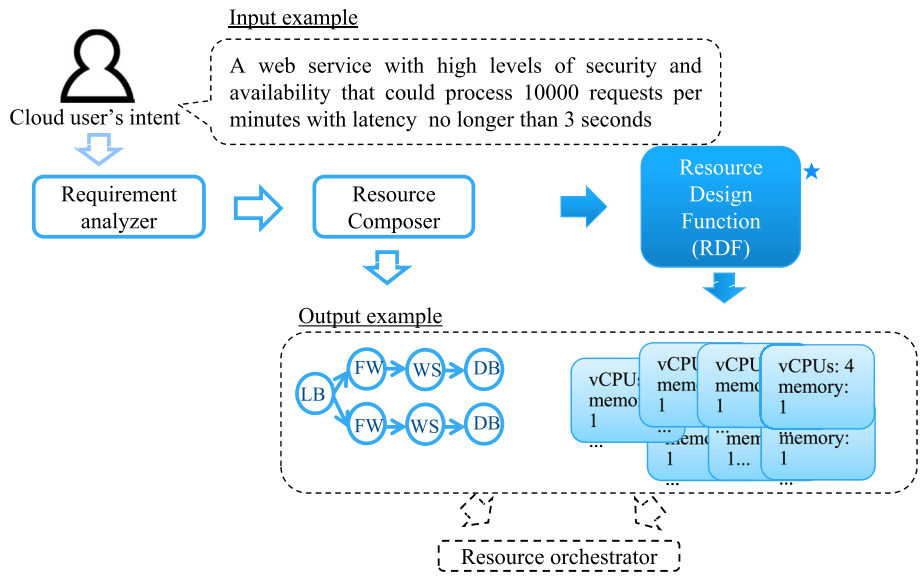

Fig. 2 Overview of Intent-based Cloud Management framework & example of input and output of ICSM. LB: Load Balancer FW: Firewall WS: Web Server DB: Database

In the current work, we focus on the realization and validation of one of the functional blocks of ICSM, RDF, which is responsible for deciding the cloud computation resource amount in accordance with the performance requirements.

To briefly introduce our preliminary work on ICSM, the input of the ICSM framework is the user's intent about the cloud service, i.e. functionality and non-functionality service-layer requirements. The ICSM framework translates the user's intent to a Resource Descriptor which is the output of the ICSM framework. The Resource Descriptor includes 2 types of resource information: (1) the information of cloud resource composition that meets the functionality requirements, i.e. service type requirement, security requirement, reliability requirement, etc., and (2) the information about the resource amount that meets the performance requirement. The Resource Descriptor could be generated in resource orchestrator-compatible format, to enable seamless deployment from intent to resource provisioning.

There are 3 main functional blocks in the ICSM framework, namely Requirement Parser, Resource Composer, and Resource Designer Function (RDF). The Requirement Parser is responsible for parsing the user's intent into atom cloud requirements related to functionality, security, reliability and performance. The Resource Composer decides the composition of the Service Function Chain (SFC) in accordance with the functionality, security and reliability requirements. The RDF is responsible for deciding the cloud computation resource amount in accordance with the performance requirement. The functional blocks can be used separately. In particular, for RDF, it could be used as a stand-alone function as long as the performance-related intent is available.

In the following part of this section, we will illustrate the detailed realization of RDF which translates the user's intent about performance into the necessary resource amount. We will firstly analyze the factors that need to be considered for resource design (section IV-1), then propose the architecture of RDF (section IV-2), and finally introduce the intent breach prevention mechanism of RDF (section IV-3).

Factors affecting decisions on resource amount

In this section, we will analyze the factors that need to be taken into consideration when deciding the cloud resource amounts, i.e., decide the amount of vCPUs, virtual memory, and storage etc. to be allocated to each instance to meet the performance intent.

(1) Workload and application performance requirements

The cloud user processes their application workload in the cloud environment, and requires the performance requirements to be met. Workload here refers to the type, features, and amount of processing. The performance requirements can be divided into two main categories: the processing time restriction and the processing percentage restriction. Table 2 gives two examples of workload and the corresponding performance requirements. For a certain workload, the computation resources allocated to the VM instances directly affect the processing time and

Table 2 Example of workload and performance requirements

Workload			Performance	
Type	**Features**	**Amount**	**Processing time restriction**	**Processing percentage restriction**
Web server	Web page size, etc.	Requests per second	Keep average process time of requests under 1 s	Successful request rate per second
Neural network	Layers, neurons, activation functions, etc.	Number of pixels, number of images, etc.	Keep training time under 10 s	...

percentage achieved. Thus, workload and performance requirements must be considered when deciding the resource amount allocated for VMs.

(2) Environmental conditions

Environmental conditions in this work are the conditions of the physical host to which the VM is allocated. Static environmental conditions include the CPU clocks and memory architecture etc. of the host; dynamic conditions include the resource usage of the host. Because the static environmental conditions are of relatively low variation and are not subject to change for a relatively long time span for a given cloud provider, we focus on dynamic environmental conditions in this work. Changes in dynamic environmental conditions such as the host's resource usage affect the performance of VMs. On the basis of this analysis, to satisfy the performance requirements, environmental conditions clearly need be considered when the resource amount is decided.

(3) Virtual machine performance requirements

The purpose of virtual machine performance requirements is to enhance the other features, e.g., the stability of the cloud-based service/application. The cloud user can optionally set virtual machine performance requirements. In this work we focus on a typical virtual machine performance requirement: VM resource usage restriction. The VM resource usage restriction restricts the resource usage inside VMs to be within a desired range. For instance, putting restrictions on the resource usage inside the VM, e.g., 50%–80%, prevents resources from being overused or underused, and thus improves the resource efficiency, avoids service interruption, and prevents bottlenecks from occurring. To meet the VM resource usage restrictions, suitable amounts of resources must be allocated to VMs.

Based on the analysis, and to design the resources in accordance with the performance-related intent, we propose an RDF whose inputs are the workload, performance requirements, and environmental conditions, and outputs are the resource configuration that meets the performance intent.

Closed-loop RDF framework

The RDF system is a closed-loop system as shown in Fig. 3, where logs about the performance, etc., are collected from the cloud environment, and knowledge about the resource and performance causal relationship is abstracted from the logs. The knowledge is then used to make decision about the resource amount configuration for the intent, and the resource configuration is implemented in the cloud environment and provided to the cloud users, and logs about the performance, etc. are collected.

RDF has three phases as shown in Fig. 3: *knowledge abstraction, decision making and implementation,* and *information collection*.

Knowledge abstraction

In the knowledge abstraction phase, the models are trained on the basis of the collected log data about workload, resource, environmental conditions and performance. The log data comes from data collected during the previous service providing period, or can be collected using active workload testing. In this phase RDF trains regression models of which the feature vector includes the workload information, the environmental condition information, and the resource configuration information, and the objective vector includes the application performance and the virtual machine performance. To prevent intent breach, which is crucial in commercial cloud service delivery, the RDF administrator needs to set RDF to N-mode or one of the P-modes, to let the model also gain knowledge about intent breach. The N-mode and P-modes are discussed in detail in the following section IV-3. The trained model, also called the hypothesis is passed to the decision making functional block.

Decision making and implementation

In this phase, first, the workload requirement, the current environment conditions, and combinations of the vCPU number and memory size are input to the hypothesis to infer the performance. Next, if the inferred application performance and virtual machine performance satisfy both the application performance requirements and the virtual machine performance requirements, the corresponding combination of resource amounts are output as resource

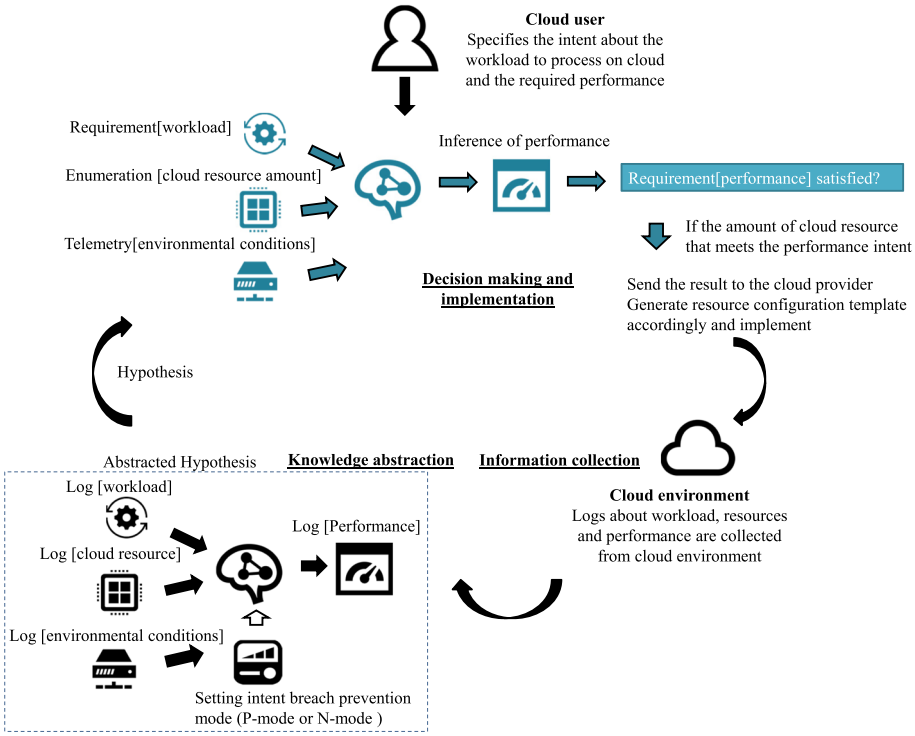

Fig. 3 RDF architecture

solutions. Finally, the resource solution is embedded in a resource orchestration template/command e.g. yaml and sent to resource orchestrators, and at the same time, the resource solution is also sent back to the cloud provider for confirmation or manual adjustment if necessary. The provider is able to confirm/revise the resource decision and instruct launching of the VMs accordingly.

Information collection phase

In accordance with the resource decision, the resource orchestrators allocate the resources and activate the VMs, and the service is provided to the cloud user. The performance and system logs such as resource usage of the host and the VM are collected and potentially used to update the models.

Intent breach prevention mechanism of RDF

In commercial cloud service delivery, intent is an important aspect of an SLA (Service Level Agreement). Intent breach happens when user's intent is not met by the provided cloud service and resources. If the intent is breached, the cloud provider needs to refund the intent breach penalty cost to the user. For instance, the cloud user and the cloud provider agree that the cloud resource must provide the cloud service with performance no worse than the performance intent *int*. If the performance becomes worse than *int*, the cloud provider needs to refund an amount of breach penalty p to the cloud user. Thus, a mechanism to prevent intent breach and so to increase the user's satisfaction is necessary, especially in commercial cloud service delivery. In this section, we will first introduce the typical intent breach penalty patterns, and then we will introduce the N-mode of RDF that adopts an existing unbiased loss function to train the model of RDF, and we will discuss its drawbacks in preventing intent breaches, i.e. the design motivation for the intent breach mechanism. After that, we will introduce the proposed intent breach prevention mechanism P-mode, and illustrate the biased loss function we propose for P-mode that is able to simultaneously retain the precision of the RDF model and lower the risk of intent breach. The validation results for N-mode and P-mode are presented in section VI.

On the basis of interviews with the cloud service operator, we find that the intent breach penalty can be categorized in the following 3 types, according to how the breach penalty p is calculated with respect to the number of times an intent breach occurs, the breach extent or the breach duration.

- Breach times penalty (BTP)
- Breach extent penalty (BEP)
- Breach duration penalty (BDP)

For BTP, each time the intent is breached, i.e., the real performance of the cloud service is worse than the

intended performance (intent), a fixed penalty α is imposed on the cloud provider, and otherwise no penalty is imposed on the cloud provider. Thus the penalty for a cloud service order is formulated as the following:

$$p = \begin{cases} 0, if\ perf\ is\ no\ worse\ than\ \ int \\ \alpha, if\ perf\ is\ worse\ than\ \ int \end{cases} \quad (1)$$

where *perf* is the real cloud performance, *int* is the performance intent, and α is a constant that specifies the penalty when an intent breach happens.

For BEP, each time the intent is breached, i.e., the real performance of the cloud service is worse than the intended performance (intent), a penalty is imposed on the cloud provider according to the percentage difference between the real performance and the performance intent, i.e., the intent breach extent. Otherwise no penalty is imposed on the cloud provider. Thus for a cloud service order, the penalty is formulated as:

$$p = \begin{cases} 0, if\ perf\ is\ no\ worse\ than\ \ int \\ \beta\ |perf - \ int|/perf, if\ perf\ is\ worse\ than\ \ int \end{cases} \quad (2)$$

where β is a constant that specifies how much penalty is imposed according to the breach extent when an intent breach happens.

For BDP each time the intent is breached, a penalty is imposed on the cloud provider according to the duration of intent breach. Thus for a cloud service order, the penalty is formulated as:

$$p = \begin{cases} 0, perf\ is\ no\ worse\ than\ \ int \\ \gamma * dur, perf\ is\ worse\ than\ \ int \end{cases} \quad (3)$$

where *dur* is the intent breach duration, and γ is a constant that specifies the how much penalty is imposed according to the breach duration when an intent breach happens.

To prevent intent breach, it is necessary to allocate sufficient resources to ensure that the real cloud performance is better than the intent. Meanwhile, from the resource efficiency perspective, allocating excessive resources is able to increase the real performance but results in a cost increase. Thus it is necessary for RDF to ensure that the real performance is better than the intent but at the same time close to the intent .

As mentioned in Section IV-2, in the knowledge abstraction phase of RDF, RDF trains a set of regression models on the basis of the collected log data. The feature vector includes the workload information, the environmental condition information, and the resource configuration information, and the objective vector includes the application performance and the virtual machine performance.

If we apply existing "unbiased" loss functions such as MAE, MSE, and MAPE to RDF as shown in (4), (5) and (6) respectively to train the model, the RDF is set in N-mode [22–24]. Note that "unbiased" here means that, for the same real performance, $perf_i$, either the real performance $perf_i$ is worse than the inferred performance h_i, or the real performance $perf_i$ is better than the inferred performance h_i, but as long as the absolute difference between the real performance and the inferred performance $|perf_i - h_i|$ is identical, the model will take it as the same loss.

$$L_{MAE} = \frac{1}{m}\sum_{O}^{i=m-1} |perf_i - h_i| \quad (4)$$

$$L_{MSE} = \frac{1}{m}\sum_{O}^{i=m-1} |perf_i - h_i|^2 \quad (5)$$

$$L_{MAPE} = \frac{1}{m}\sum_{O}^{i=m-1} |perf_i - h_i|/perf_i \quad (6)$$

However, in the case of service delivery, the inference may lead to different risk level of intent breach, even if the absolute difference between the real performance and the inferred performance $|perf_i - h_i|$ is the same, depending on whether the real performance $perf_i$ is better or worse than the inferred performance h_i. We will give an example to illustrate this. For example, assume that the cloud user has an intent to encode a 100GB in 10 s using a given encoder that runs on a VM, and would like to know how much computation resource is needed to be allocated to the VM to meet the intent. Let's consider 2 situations where the absolute difference between the real performance and the inferred performance is the same:

(1) The inferred performance is better than the real performance. Assume that the RDF model infers that for the given amount of workload, i.e., encoding a 100GB video, allocating 4 vCPUs to the VM to process the workload will result in a task taking 9.9 s (inferred performance h_i), while the real performance is 10.1 s. In this case, when RDF used the model to determine resources, it decided that 4 vCPUs were sufficient to meet the intent and instructed the resource orchestrator to implement the service with 4 vCPUs. However, in implementation, allocating 4 vCPUs to the VM to process the workload took 10.1 s (real performance), so intent breach happened since that the real performance was worse than the intended performance of 10 s.

(2) The real performance is better than the inferred performance. Assume that the RDF model infers that for the given amount of workload, i.e., encoding a 100GB video, allocating 4 vCPUs to the VM to process the workload will result in a task taking 10.3 s (inferred performance h_i) while the real performance is 10.1 s. In this case, when RDF used the model to determine resources, it decided that 4vCPUs were unable to meet the intent and searched for other available resource design solutions. Thus prevented the intent breach happening.

In both the 2 cases described above, the absolute difference between the real performance and the inferred performance is the same at 0.2 s. However, while in case (1) the inferred performance is better than the real performance, and resulted in a higher intent breach risk, in case (2) the inferred performance is worse than the real performance, and resulted in a lower intent breach risk.

This observation indicates the N-mode loss functions' drawbacks in distinguishing inferences that lead to different risk levels of intent breach. On the basis of the observation, we have proposed the P-mode. P-mode is able to lower the intent breach risk by adopting our proposed loss functions L_{BTP}, L_{BEP}, L_{BDP}. These loss functions impose a penalty on performance inferences that lead to a high risk to intent breach.

On the basis of the breach penalty patterns, we have proposed 3 models for P-mode, which are P_{BTP}, P_{BEP}, P_{BDP}. The loss functions of P-mode models (7), (9), and (10) below, are composed of 2 parts. The first part is the unbiased loss function, for which, in this work, we use MSE. However, when the first unbiased loss function is MAE or MAPE, the P-mode loss function could be formulated in a similar way. The second part is the penalty function which is imposed on the total loss function when the inferred performance penalty h_i is better than the real performance $perf_i$.

For P_{BTP}, the cloud provider sets RDF mode in the knowledge abstraction phase to P_{BTP} mode, and so during the learning process, the loss function L_{BTP} is applied:

$$L_{BTP} = \frac{1}{m}\sum_{O}^{i=m-1}(|perf_i - h_i| + \varepsilon * b_i) \tag{7}$$

and,

$$b_i = \begin{cases} 0, perf_i \text{ is no worse than } h_i \\ 1, perf_i \text{ is worse than } h_i \end{cases} \tag{8}$$

where m is the number of training data records, and $perf_i$ is the real performance value for ith training data records, h_i is the inferred performance value of ith training data, and ε is an adjustable constant. As we can see from the formula of L_{BTP}, a fixed penalty is added to the absolute error between the real performance and inferred performance when the real performance is worse than the inferred performance.

For P_{BEP}, the cloud provider sets the RDF mode in the knowledge abstraction phase to P_{BEP} mode, and so during the learning process, the loss function L_{BEP} is applied:

$$L_{BEP} = \frac{1}{m}\sum_{O}^{i=m-1}(|perf_i - h_i| + \varepsilon * b_i * |perf_i - h_i|) \tag{9}$$

As we can see from the formula of L_{BEP}, the weighted difference between the real performance and the inferred performance is added to the absolute error between the real performance and inferred performance when the real performance is worse than the inferred performance.

For P_{BDP}, the cloud provider sets the RDF mode in the knowledge abstraction phase to P_{BDP} mode, so that during the learning process, the loss function L_{BDP} is applied:

$$L_{BDP} = \frac{1}{m}\sum_{O}^{i=m}(|perf_i - h_i| + \varepsilon * b_i * dur_i) \tag{10}$$

where dur_i is the duration of intent breach for the ith training data. As we can see from the formula of L_{BDP}, the weighted intent breach duration is added to the absolute error between the real performance and inferred performance when the real performance is worse than the inferred performance.

In the case where the intent is process time restriction, when the required process time restriction is not met, the intent breach duration dur_i equals the real process time, i.e. $perf_i$. Thus in this case, the loss function for L_{BDP} is rewritten as,

$$L_{BDP} = \frac{1}{m}\sum_{O}^{i=m}(|perf_i - h_i| + \varepsilon * b_i * perf_i) \tag{11}$$

To conclude, RDF has two modes: Normal mode (N-mode) and intent breach Prevention mode (P-mode). N-mode is the baseline mode in which no intent breach prevention mechanism is applied. The objective of P-mode is to enhance the service quality and user's satisfaction by suitably adding bias to performance inference to decrease intent breach risk while ensuring high inference accuracy of performance.

Application of RDF to a sensor-cloud system

In this work, we have applied RDF to a sensor-cloud system as shown to validate RDF. In this section, first we will illustrate the demand for RDF of a sensor-cloud system (V-1), then we will introduce the experiment setup to simulate the sensor-cloud system and illustrate how the performance-related logs are collected from the simulated sensor-cloud system (V-2).

RDF for a cloud-sensor system

Sensor-cloud systems are widely used in various scenarios such as smart cities, smart agriculture, autonomous security monitoring and analysis of public facilities, autonomous disaster monitoring and detection, remote medical care, etc. In a cloud-sensor system, numerous sensors and smart devices generate enormous amounts of data at a high rate, and it is usually difficult for humans to analyze all the data and make decisions. Thus to utilize the data, most of the systems adopt learning mechanisms to continuously analyze the data and conduct decisions or provide assistance in decision making. Central cloud infrastructures provide high computation ability and are usually used to implement the learning mechanism. In most of the cases, it is essential to adapt to new situations in the scenarios, continuously learning from the enormous amount of collected data on the cloud.

For the operator/administrator of the sensor-cloud system, it is important to carefully allocate sufficient cloud resource to ensure that the learning program's process of training data implemented on the cloud does not fall behind the generation speed of newly collected training data, otherwise, there will be a growing queue of newly generated training data waiting to be used for training, and the learning mechanism will not be able to capture the new situations in real time. For instance, in a sensor-cloud system, if the system collects 100 sets of training data per second, while the computation ability provided by the central cloud to the learning mechanism is only able to train the model on 20 sets of training data, there will be a growing stock of un-trained data, and the features in the newly collected un-trained data will not be reflected in the model in real time. In other words, the challenge for the cloud user of a sensor-cloud system is that, when using the cloud to train on continuously collected data, it is necessary to ensure that it takes no more than a given execution time restriction to train on a certain amount of data using a certain model in a given time restriction.

However, it is an extremely skill-demanding process for the operator/administrator of the sensor-cloud system to determine the necessary amount of cloud resource to meet the training time restriction for various learning model variations, training set features, etc. To resolve the challenge, we have applied RDF to this scenario. In this validation experiment, we focus on the typical sensor-cloud system where the collected data is image data, and the learning mechanism on the central cloud is a neural network based learning mechanism (Fig. 4). To give an example of the scenario, in a smart transport system, cameras (sensors) set in the transport system collect images of the current traffic situation, and on the basis of the collected images, information including the presence of cars, bikes and pedestrians is abstracted utilizing image recognition and a machine learning mechanism. On the basis of this information, the transport system administrator (system or human) optimizes the transport system, e.g. traffic light duration, etc. To adapt to new traffic situations, the image recognition model needs to be updated frequently using new collected images. RDF in this scenario is aimed at determining the necessary cloud resource to meet the training time requirements for various neural network based learning models and various image training data sets of different sizes.

Experiment setup

To show the application of RDF to the sensor-cloud system scenario, we have constructed an experiment environment for the sensor-cloud system as shown in Fig. 5. The experimental cloud platform is used to collect the log data, which is used to train the models in RDF.

The summaries of hardware and software used in the experiment are shown in Table 3 and Table 4 respectively.

In the experiment, to collect the log data for the aforementioned sensor-cloud system where the collected sensor data is images and the learning mechanism deployed on the central cloud is a neural network, we have implemented a cloud computation node and a set of tools (Table 4). The performance log data (the execution time of the neural network) is collected under the conditions of using various resource amounts (number of vCPUs, memory), image data sets of various numbers of pixels,

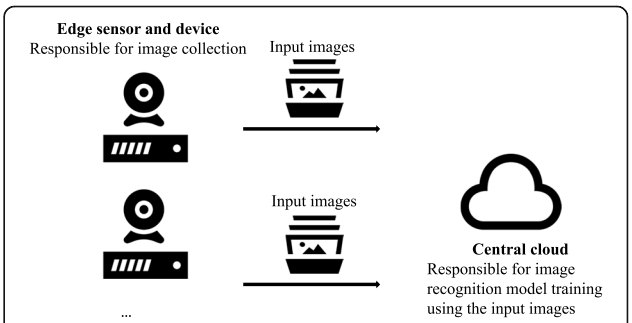

Fig. 4 Validation scenario: a cloud-Sensor System where the collected data is image data, and the learning mechanism on the central cloud is a neural network based learning mechanism

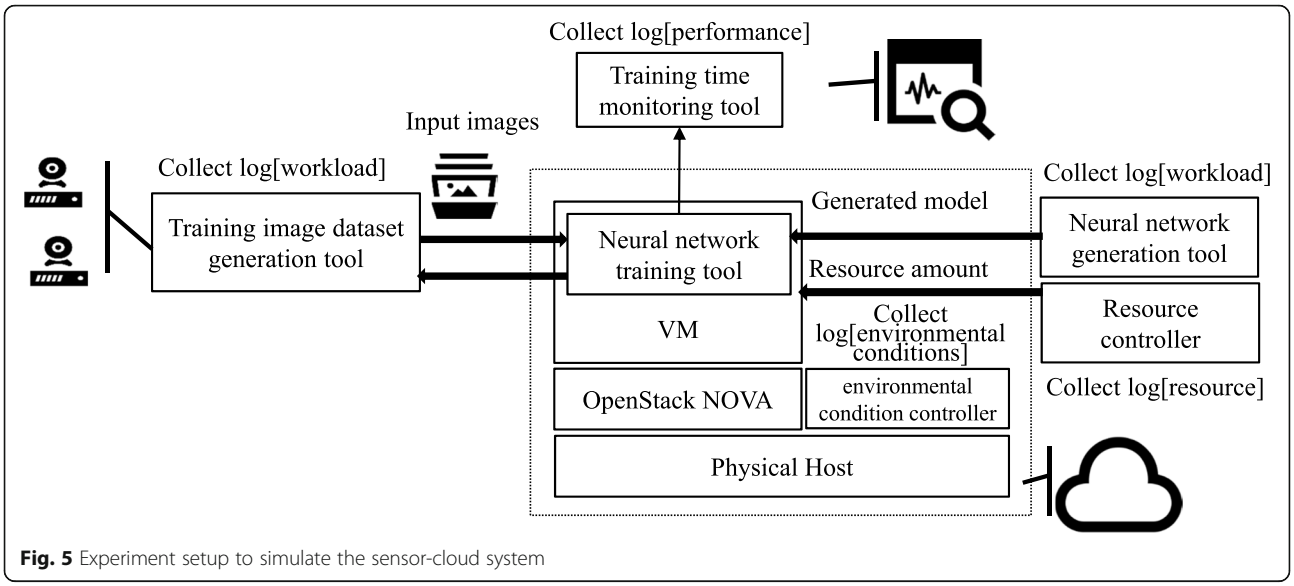

Fig. 5 Experiment setup to simulate the sensor-cloud system

and color channels for training neural networks of various layers, neurons, and activation function. In other words, we keep the resource amount and workload features (image features and neural network features) variable, and record the performance data for the different combinations.

In the following part of this section, we will first introduce the procedure and parameter settings used to collect the log data, which is automated by the log data collection workflow controller (also called workflow-C, implemented in Python). We will then introduce the tools that the workflow-C takes control of; these tools are used to generate image datasets and the neural network, to control the resource amount, to monitor the performance, etc.

The workflow controller is implemented in Python to enable autonomous data collection. Figure 6 shows the log data collection workflow controller's process for collecting the log data.

As shown is Fig. 6, the workflow to collect the data is as follows.

There are external loops and internal loops to collect the data. In each external loop, the resource amount for the VM and the environment conditions is set for the internal loops contained in the loop. In each internal loop, the workload (the image data set, the neural network) is set, and the neural network is trained using the image dataset with the given amount of cloud resource, and the performance is recorded. Each external loop is repeated for LE times and each internal loop is repeated for LI times. The detailed operations in the loop is as follows (Fig. 6).

Note that in the experiment, values of the resource amount, values of environmental conditions, the workload (the image training dataset parameters and the image recognition models) are chosen randomly each time with equal possibility (uniform distribution) in the range as shown in Table 5 to simulate a wide variety of scenarios and collect comprehensive log data.

(1) If the external loop has been repeated fewer than LE times, continue to (2); if not, end the collection.
(2) As the external loop starts, the workflow controller randomly selects the resource amount allocated to the VM. In the experiment, the number of vCPUs allocated to the VM is selected from the range 1 ~ 8 with equal possibility as shown in Table 5, and the size of vMemory allocated to VM is selected from the range 16 ~ 64GB with equal possibility. Then the workflow-C instructs the resource controller to allocate the resource and activate the VM.
(3) As the internal loops starts, the workflow-C instructs the training image generation tool to generate randomly an image dataset according to the range in Table 5 with equal possibility, and instructs the neural network generation to randomly generate a random image recognition model from the range shown in Table 5. The image recognition models are trained using the generated dataset on the VM

Table 3 Hardware specifications of central cloud servers

Specifications	Value
Model	Fujitsu PRIMERGY RX2530 M2
CPU	Xeon 5 E-2509v 8core
Memory	128 GB
Disk	1 TB

Table 4 Software used in the validation experiment

Software used	Role of the software
Log data collection workflow controller (also called workflow-C)	To take control of log data collection workflow and instruct other controllers (described below and in section V-2) to enable autonomous data collection
Openstack	Virtualization infrastructure management
KVM	Hypervisor of the cloud
Resource controller (scratch)	To control the resource amount allocated to the VM according to the workflow-C′ instruction
Environmental condition controller (Stress-ng [25])	To simulate the environmental conditions of the host according to the workflow-C′ instruction
Training image dataset generation tool (scratch)	To generate a training data set of a given number of images with a given number of pixels and color channels according to the workflow-C′ instruction
Neural network generation tool (based on Tensorflow [26]))	To generate image recognition models of given layers, given number of neurons in each layer and activation functions according to the workflow-C′ instruction
Neural network training tool (based on Tensorflow [26])	To train the generated model with the generated image dataset using the given amount of resource on the VM
Training time monitoring tool (based on Tensorflow [26])	To record the execution time of for the generated model and training data set using the given amount of resource on the VM

Tools marked scratch is built from scratch

for 5 times, and the average training time (also called execution time in this paper) is recorded. After the training process, the log data about the resource amount (log [resource]), environmental conditions (log [environment]), workload (log [workload]), and performance (log [performance]) are aggregated as one set of log data. The columns of the log data are as shown in Table 6.

(4) If this internal loop has been repeated fewer than LI times, go back to (3); if not, go back to (1).

We have set the number of external loop LE (Loop External) to 500 and the number of internal loop LI (Loop Internal) to 50 and collected around 23,000 sets of valid log data in this experiment.

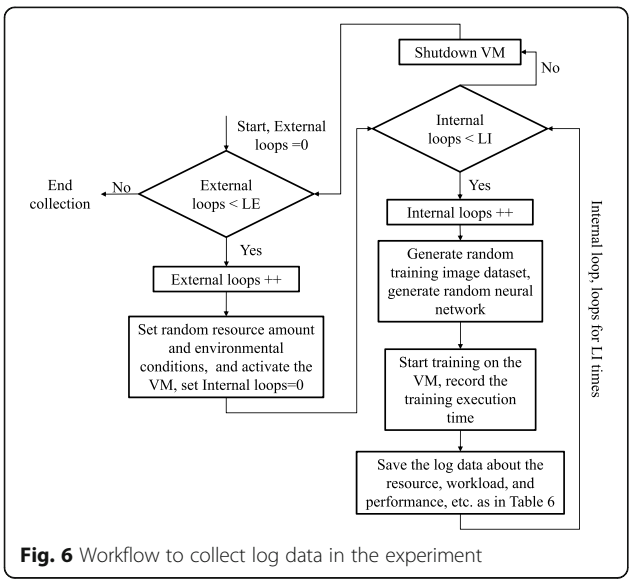

Fig. 6 Workflow to collect log data in the experiment

We have also implemented several local controllers (Table 4) to control the workload patterns, resource patterns and collected log data under these conditions. All these controllers take instructions from workflow-C, and control the resource amount, generate the workload (the training dataset and neural network) in accordance to the instructions. As mentioned above, the values for these parameters are chosen randomly with equal possibility (uniform distribution) in the range as shown in Table 5. The different controllers categorized by the functions are as follows:

(1) To control the central cloud environment and cloud resource variations:

On the computation node, we have implemented a resource amount controller that is responsible for allocating various patterns of computation resources to VMs. We have also implemented an environmental condition controller that is responsible for generate background resource usage including host CPU usage, memory usage to simulate the environmental conditions. The controller is built based on Stress-ng [25] which is able to stress the resource usage of a host to a given percentile.

(2) To generate the training images dataset (the workload)

We have implemented a training image dataset generation tool using Python to generate various training image datasets of random pixels, random color channels, and of a random size. It is implemented from scratch based on python

Table 5 Random parameter range of workload, resource and environmental conditions in the experiment

	Parameter settings	Random value range
Training image data set variation	Number of training images	[1 ~ 30]*1000
	Training image pixels	horizontal[4 ~ 64], vertical [4 ~ 64]
Image recognition model variation	Number of dense layers	1–20
	Number of neurons for each dense layer	1–50
	Number of convolutional layers	1–20
	Number of neurons for each convolutional layer	1–50
	Activation function for each layer	softmax, relu, sigmoid
Resource amount range	Number of vCPUs allocated to VM	1 ~ 8
	Size of vMemory allocated to VM	16 ~ 64GB
Environmental conditions range	CPU usage of host	0 ~ 100%
	Memory usage of host	0 ~ 100%

(3) To generate the image recognition model and train it on the central cloud

We have implemented a neural network generation tool using Python and Tensorflow [26] to generate various image recognition models. The models are based on neural networks, which are composed of different numbers of convolution layers and dense layers, and different numbers of neurons and activation functions for each layer.

The generated training image dataset and the generated image recognition model are input to the neural network training tool (implemented using Python and Tensorflow [26]), and the model is trained on the VM with the given amount of resource.

(4) To monitor the performance

A training time monitoring tool is implemented on the basis of Tensorflow [26] to record the execution time to train the generated image dataset using the generated neural network on the VM with the given amount of resource.

RDF validation results

As introduced in Sections I and II, the objective of RDF is to resolve complexity in translating a cloud user's intent into a concrete cloud resource configuration that satisfies the intent, so reducing the human and time cost involved in the resource design to meet the user's intent, while at the same time ensuring high precision in

Table 6 Keys and descriptions of collected log data, i.e. training data for RDF models

	Key	Description
Log [resource]	*vcpu*	the number of vCPUs allocated to the VM
	vmemory	the memory size allocated to the VM
Log [environment condition]	*host_mem_usage*	the memory usage of the host
	host_vCPU_usage	the CPU usage of the host
Log [workload]	*image_pixels*	the horizontal and vertical pixels of the input image
	convlayer_num	the number of convolution layers
	conv1_neurons	the number of neurons in the first convolution layer
	....	....
	Conv20_neurons	the number of neurons in the 20th convolution layer
	denselayer_num	the number of dense layers
	dense1_neurons	the number of neurons in the 1 st dense layer
	....	....
	Dense20_neurons	the number of neurons in the 20th dense layer
	conv_activation	the activation function of the convolution layers
	dense_activation	the activation function of the dense layers
Log [performance]	*excutime*	the average training time

resource design. Thus, to evaluate the effectiveness of RDF, we have evaluated it to see whether RDF is able to design resource precisely with less human and time cost compared to the conventional resource design approach. Furthermore, we have proposed intent breach prevention mechanisms (P_{BTP}, P_{BEP}, and P_{BDP} for RDF) to reduce the intent breach risk, thus ensuring high user satisfaction. Thus in this chapter, we will firstly evaluate the effect and advantage of RDF over the conventional resource design approach by evaluating the RDF's precision and the human and time cost compared to the conventional resource design approach (VI-1); then we will evaluate the effectiveness of RDF's intent breach prevention mechanism (VI-2).

Evaluation of effectiveness of RDF

In this subsection, we will evaluate the effectiveness and advantage of RDF over the conventional resource design approach by evaluating the RDF's precision and the human and time cost compared to the conventional resource design approach.

As introduced in section IV, in the knowledge abstraction phase, RDF trains a model which is able to infer the performance for given resource configurations, workload and environmental conditions. The model is then used in the decision making phase to infer the performance for all the available resource configurations, and on the basis of the inferred performance, a resource configuration that meets the performance requirements specified by the intent is chosen as the output of RDF. Thus, the performance inference precision is the key factor that decides the resource design precision of RDF. Figure 7 shows an example of a resource design result produced by RDF. For a given user intent, RDF infers the performance (horizontal axis), i.e. training execution time in the validation scenario for each available resource combination (vertical axis). The blue vertical line in the figure shows the performance intent i.e. the execution time restriction requirement-training time no longer than 0.05 s. For each inferred performance that meets the intent, the corresponding resource amount combination (memory/vCPU) is outputted as a recommended resource amount combination for the intent. The user or operator may further choose from the recommended resource amount combinations on the basis of other strategies, e.g. choose the resource amount with the lowest cost. From the example, we can see that the performance inference precision is the key factor that influences the correctness of the resource design by RDF. We will illustrate the performance inference precision evaluation in the following paragraphs.

We trained RDF N-mode and P-mode models using the 23,000 sets of log data collected in the experiment. We used neural network regression to train the RDF models. The reason for using neural network regression is that thanks to rehearsal experiments, it was observed that the relationship between the model input and output is non-linear, and neural network based models surpass other models (e.g., linear regression, polynomial regression) in performance inference precision. To find the optimal model design for N-mode and P-mode, we have used 5-fold validation to calculate the precision of inference precision. To calculate the precision under 5-fold cross-validation, the log data is split into 5 sets randomly, and for the first iteration, the first set is used to test the model and the rest are used to train the model. In the second iteration, the second set is used as the testing set while the rest serve as the training set. This process is repeated until each of the 5 sets has been used as the testing set. The average of the precision of the 5 iterations is then calculated as the precision of the model.

Furthermore, we have performed a grid search of neural network parameters including layers, neurons, optimizer, and activation function to find the optimal parameter settings that maximize the precision of each N-mode and P-mode model. (The range of grid search is omitted due to limited space.).

RDF's performance inference result of one set of test data is shown in Fig. 8. The horizontal axis is the real (observed) performance and the vertical axis is the inferred performance. We can see that for the N-mode model and P-mode model, the data points converge to the line where the inferred performance and the real performance are equal.

Figs. 9, 10 and Table 7 show the statistical precision evaluation result of performance inference by RDF. We have evaluated the average MAPE, MAE and the precision of the 5-fold cross-validation.

The performance inference precision is defined as the following,

$$Precision = \frac{\sum_{i=1}^{m} 1 - \left| ext_i^{real} - ext_i^{predicted} \right| / ext_i^{real}}{m} \quad (9)$$

where m is the number of evaluation data sets, ext_i^{real} is the ith observed execution time, and $ext_i^{predicted}$ is the ith inferred execution time of the machine learning in the evaluation data set for the given resource configuration, workload and environmental conditions.

Table 7 shows the precision evaluation results for RDF's N-mode model (N_{MAPE} is the N-mode model trained with the conventional loss function MAPE) and the P-mode model. For the evaluation matrix MAPE, the RDF models have achieved MAPE between 9.3707% and 11.0672%, among which N_{MAPE} has performed the best.

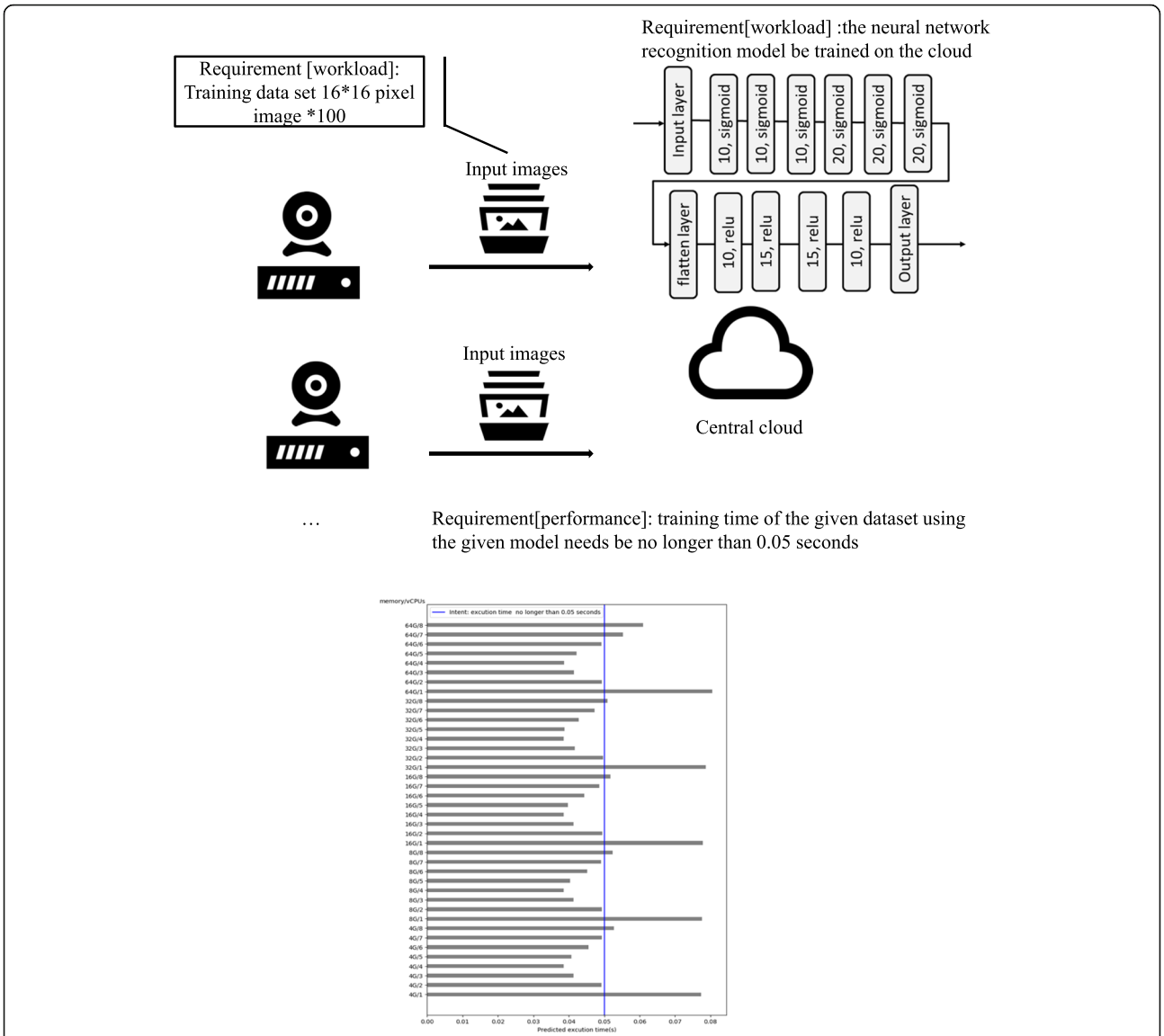

Fig. 7 An intent example of the application engineer for the sensor-cloud system (above) and the performance inference and Resource Design for the intent example by RDF

For the evaluation matrix MAE, the RDF models have achieved MAE between 0.0209 and 0.0282 s, where P_{BTP} performed the best. For the evaluation matrix, the models achieved 89.1031–90.6293 precision, and N_{MAPE} performed the best. That means that for a given set of workload, resource combination, and environmental conditions, the mean percentage difference between the inferred performance and real performance is no more than 11.0672%. Thus on the basis of the models, the resources were designed precisely.

To evaluate the advantage of RDF, we want to compare it with the conventional resource design approach. According to our investigation of related research (section III), since to the best our knowledge, translation of cloud performance-related intent into cloud resource amount has barely been studied, we have compared RDF with the resource design approach currently used in the cloud industry. As introduced in section II, according to our interviews with cloud operators, currently there are two main approaches to translating the user's intent about the cloud service ("what") into "how" the service is implemented: the *cloud-consultant approach* and the *self-service approach.* In both approaches, translation of service layer requirements into resources relies heavily on human decisions. We have compared the RDF with the conventional human decision-based design approach from the time cost, human resource cost and resource design result aspects.

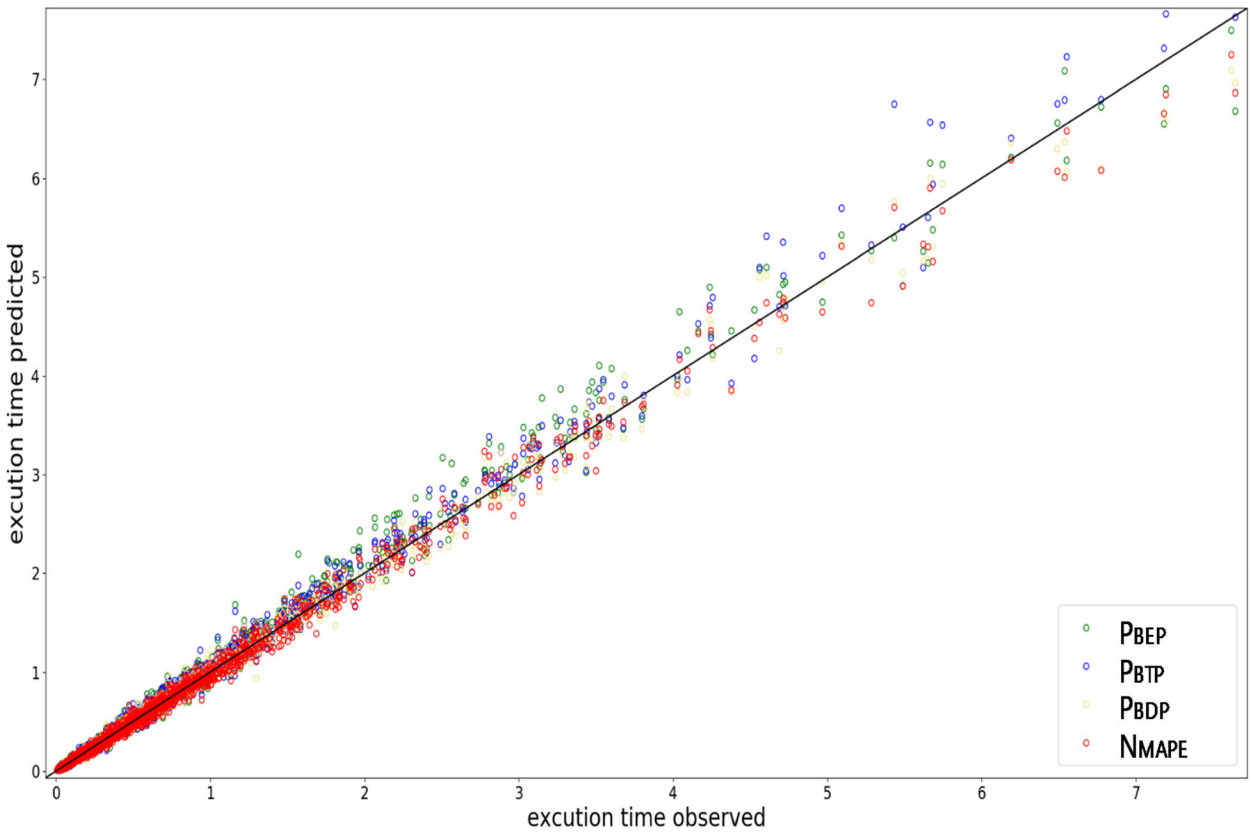

Fig. 8 Performance inference result of rdf: comparison between observed execution time and inferred execution time

For the time cost, since RDF infers the performance for all available resource configuration patterns and chooses the resource configuration that meets the intent, the time consumed to design the resource is dependent on the number of available resource configuration patterns. Figure 11 and Table 8 show the average resource design time against the number of resource configuration patterns. We can see from the results that the resource design time of RDF increases as the number of resource amount variations increases, and the relationship between the two can be approximated as linear. For commercial cloud services such as AWS EC2, the available resource configurations, also called instance types, are usually limited to tens to hundreds of variations, and according to the results (Fig. 11 and Table 8), RDF is able to find the resource configuration to meet the performance requirement in 2.11 s (RDF execution time for 1000 resource amount variations) for a given workload.

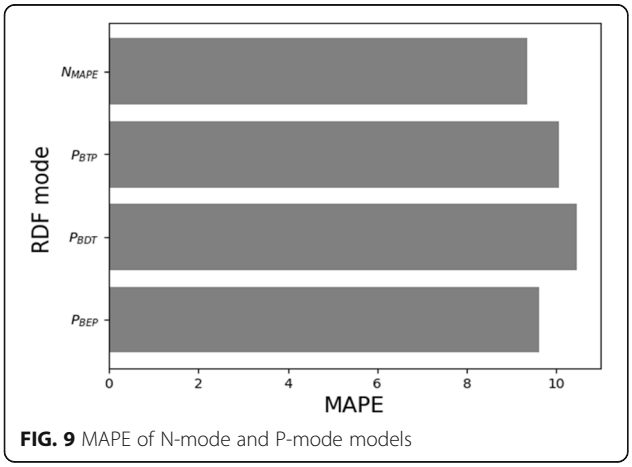

FIG. 9 MAPE of N-mode and P-mode models

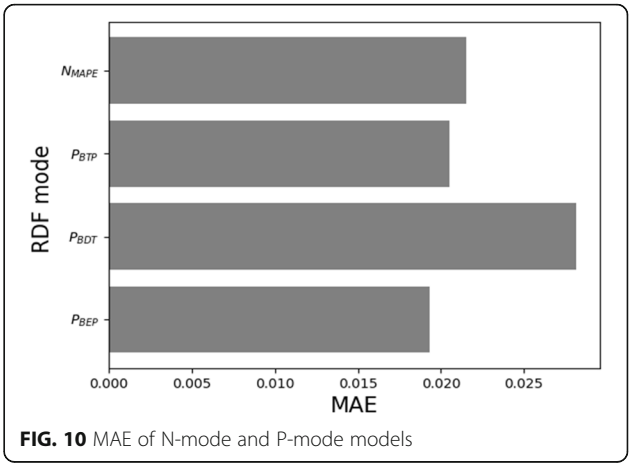

FIG. 10 MAE of N-mode and P-mode models

Table 7 Precision of n-mode and p-mode models

RDF models	MAPE(%)	MAE	Precision (%)
P_{BEP}	10.8969	0.0217	89.1031
P_{BDP}	10.4655	0.0282	89.5345
P_{BTP}	11.0672	**0.0209**	88.9328
N_{MAPE}	**9.3707**	0.0219	**90.6293**

The best peformance for each evalution matrix is marked as bold.

As for the conventional human-based resource design approach, the design time largely varies according to the designer's skill level and other factors, and, according to our interviews, usually hours to days is needed to design the cloud resource for a human-based design approach. Thus RDF largely surpasses the conventional human-based resource design approach in the aspect of time cost by reducing resource time from hours-days to seconds.

For the human resource cost, since the design of the cloud resource requires the designer to have knowledge and experience about the cloud platform and the relationship between workload, resource and performance, it results in a high human cost for the cloud provider or cloud user. RDF automates the resource design by utilizing the knowledge (model) abstracted from history log data, so, as long as models are built for RDF, only minor human cost is demanded in the resource design process. Thus the application of RDF for cloud resource design could greatly save human cost compared to the conventional approach. Note that to implement RDF, an interview with the targeted application's administrator is necessary to identify the types of performance requirement, and this process requires human involvement. Nevertheless, training the designers for the task is necessary for the conventional human-based approach, and this usually takes more human cost than the interview human cost for RDF.

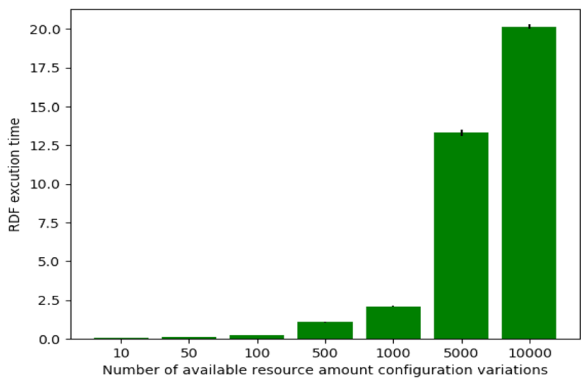

Fig. 11 RDF execution time for different numbers of resource amount variations

For the resource design result, RDF enumerates all available resource amount configurations, and infers the performance for each available configuration and thus is able to output all the configurations that meet the performance requirement. It would be challenging for conventional human-based resource design to find all resource configurations that meet the performance requirement. As for the precision of the resource design result, as mentioned before (Table 7), RDF achieves at least 89.1% precision in inferring the performance for given workload and resource configuration; in other words, RDF is able to ensure that the designed resource is able to ensure that the error between the performance it promises to achieve and the performance it actually achieves is on average 10.9%. In contrast, for the human-based conventional resource design approach, the difference varies largely according to the skill of the resource designer.

To conclude (Table 9), compared to conventional resource design approaches, RDF is able to shorten the resource design time from hours-days to seconds-minutes, to significantly reduce the human cost necessary by automating the resource design process utilizing the knowledge abstracted from log data, and to find all resource amount variations that meet the user's intent. As a result, it enables the user to choose the optimal resource design result in accordance with price policy, etc.

Evaluation of intent breach prevention mechanism in RDF

To evaluate the intent breach prevention mechanism P-mode of RDF, we have designed 3 evaluation functions, namely $E_{breach\text{-}times}$, $E_{breach\text{-}extent}$, $E_{breach\text{-}duration}$.

To evaluate how the RDF model performs in inferring performance that leads to intent breach, from the aspect of whether a breach happens, we use the evaluation metric $E_{breach-times}$. $E_{breach-times}$ is a value between 0 and 1; the higher $E_{breach-times}$ is, the more likely it is that an intent breach will happen.

Table 8 Analysis of RDF execution time for different numbers of resource amount variations

Number of available resource amount variations	Mean RDF execution time	Standard derivation
10	0.0745	0.000502111
50	0.1502	0.001627085
100	0.2566	0.004818735
500	1.0813	0.011302444
1000	2.1073	0.02394739
5000	13.302	0.18511307
10,000	20.1451	0.148079558

Table 9 Effect of RDF compared to conventional resource design approach

		Conventional Resource design Approach	RDF
Human resource cost	Human experience with cloud design	Necessary	Not necessary
Time cost	Design time	Hours-days	Seconds-minutes
Resource design results	Whether able to find all resource variations that meets the requirements	Difficult	Yes
	Optimal resource design	Difficult	Yes

$$E_{\text{breach-times}} = \frac{\sum_{i=1}^{m}(b_{bool})}{m} \tag{10}$$

$$b_{bool} = \begin{cases} 0, \text{if } ext_i^{real} \leq ext_i^{predicted} \\ 1, \text{if } ext_i^{real} > ext_i^{predicted} \end{cases} \tag{11}$$

To evaluate how the RDF models perform in inferring performance that leads to intent breach from the aspect of the extent to which the intent is breached, we use the evaluation metric $E_{\text{breach - extent}}$.

$$E_{\text{breach-extent}} = \frac{\sum_{i=1}^{m}(b_{extent})}{m} \tag{12}$$

$$b_{extent} = \begin{cases} 0, \text{if } ext_i^{real} \leq ext_i^{predicted} \\ \dfrac{ext_i^{real} - ext_i^{predicted}}{ext_i^{real}}, \text{if } ext_i^{real} > ext_i^{predicted} \end{cases} \tag{13}$$

To evaluate how the RDF models performs at inferring performance that leads to intent breach from the aspect of the length of time that the intent is breached, we use the evaluation metric $E_{\text{breach - duration}}$.

$$E_{\text{breach-duration}} = \frac{\sum_{i=1}^{m}(b_{duration})}{m} \tag{14}$$

$$b_{duration} = \begin{cases} 0, \text{if } ext_i^{real} \leq ext_i^{predicted} \\ ext_i^{real}, \text{if } ext_i^{real} > ext_i^{predicted} \end{cases} \tag{15}$$

Table 10 Figs. 12, 13 and 14 show the intent breach risk $E_{\text{breach-times}}$, $E_{\text{breach-extent}}$, $E_{\text{breach-duration}}$ of P-mode models P_{BTP}, P_{BEP}, P_{BDP} compared with the baseline N-mode model N_{MAPE} which applies a MAPE loss function. From the result we can see that P-mode models are able to reduce the intent breach risk with regard to breach times, breach extent, and breach duration, compared to the baseline N_{MAPE}. The details are as follows:

For breach extent risk $\mathbf{E_{breach\text{-}extent}}$, all of the three P-mode models outperform N-mode models, and P_{BTP} performs the best, achieving a reduction of 45.25% compared to the baseline N-mode model. For breach duration risk $\mathbf{E_{breach\text{-}duration}}$, all of the three P-mode models outperform N-mode models, and P_{BDP} has realized a significant reduction of 66.05% compared to the baseline N-mode model. For breach times risk $\mathbf{E_{breach\text{-}times}}$, and P_{BTP} performed the best, achieving a reduction of 43.02% risk compared to the baseline model.

At the same time, P-mode models maintain almost the same level of precision compared to the baseline model (Table 7). For performance inference MAPE, the highest increase in MAPE is witnessed in P_{BTP} which increases the MAPE to 11.0672% compared with the N_{MAPE} for which MAPE is 9.3707%. For performance inference MAE, the highest increase in MAE is witnessed in P_{BDP} which increases MAE to 0.0282 compared with the N_{MAPE} of which MAE is 0.0219; P_{BEP} and P_{BEP} even reduce the MAE by a small extentFor performance inference precision, the highest decrease of precision is witnessed in P_{BTP} which reduces the precision to 88.9328% compared with N_{MAPE} for which the precision is 90.6293%. Thus, we can state that P-mode models P_{BTP}, P_{BEP}, and P_{BDP} significantly reduce the intent breach risk at the cost of a relatively small decrease in inference precision.

Table 10 Intent breach risk evalution of n-mode and p-mode models

	$E_{\text{breach-extent}}$	$E_{\text{breach-duration}}$	$E_{\text{breach-times}}$	Reduction of $E_{\text{breach-extent}}$ compared with N_{MAPE}	Reduction of $E_{\text{breach-duration}}$ compared with N_{MAPE}	Reduction of by $E_{\text{breach-times}}$ compared with N_{MAPE}
P_{BEP}	**0.031**	0.112	0.3664	**45.25%**	45.61%	36.72%
P_{BDP}	0.0438	**0.0699**	0.402	22.41%	**66.05%**	30.58%
P_{BTP}	0.0333	0.1206	**0.3299**	40.95%	41.44%	**43.02%**
N_{MAPE}	0.0564	0.206	0.579	–	–	–

The best peformance for each evalution matrix is marked in bold

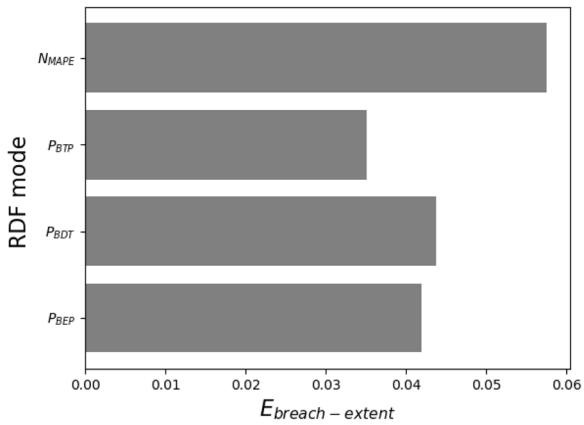

Fig. 12 Ebreach-extent of N-mode and P-mode models

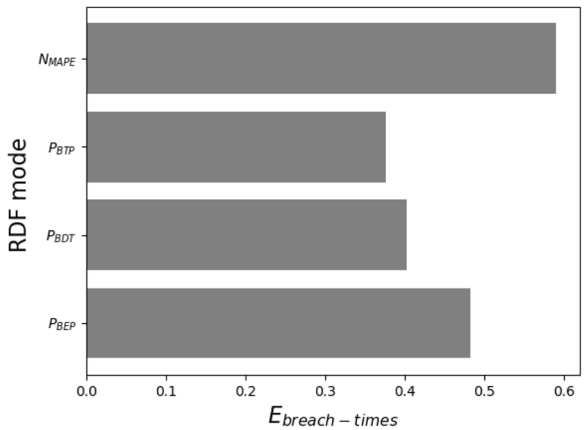

Fig. 14 Ebreach-times of N-mode and P-mode models

Furthermore, it is observed from the results that the specific P-mode model designed on the basis of a certain intent breach penalty pattern naturally outperforms other P-models for the intent breach risk evaluation matrix that is based on the same intent breach pattern. For instance, P_{BTP} which is designed on breach time penalty patterns outperformances the other P-mode and N-mode models when the evaluation matrix is $E_{breach-times}$. Thus, the administrator of RDF could easily choose the corresponding P-mode in accordance with the intent breach penalty pattern agreed between the cloud provider and cloud user in commercial cloud service delivery.

Conclusion and future work

In this work, we have addressed the gap between the cloud user's service-level requirements, also called the cloud user's intent in this work, and the cloud provider's concerns, such as the necessary resource amount to meet the intent, and have introduced the current human skill-based approaches to "translate" user's intent to resource amount and the drawbacks of such approaches. Next, we introduced our investigation of related work; we found that although Intent-based Management has been applied in various scenarios including SDN, 5G, and traffic management, barely any work has been done on realizing intent-based cloud management. To resolve the challenge of translating user's intent into resource configurations, we have proposed an Intent-based Cloud Service Management (ICSM) framework, and focused on the design and realization of the Resource Designer Function (RDF) which translates the cloud user's intent about service performance into resource amount, which the cloud provider is concerned with. Furthermore, to lower the risk of intent breach, which is crucial in commercial cloud service delivery, we have proposed an intent-breach prevention mechanism. To validate the effectiveness of RDF, we have applied RDF to a sensor-cloud system scenario.

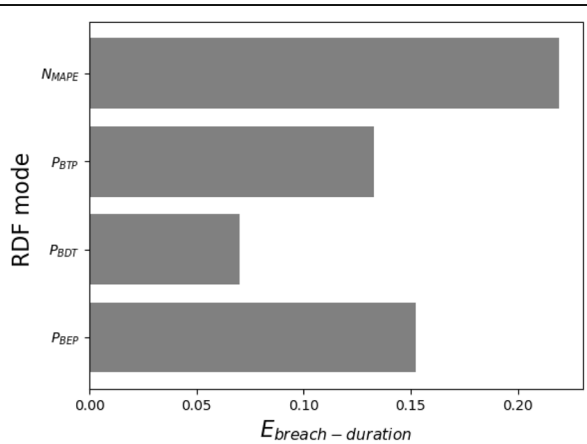

Fig. 13 Ebreach-duration of N-mode and P-mode models

From the evaluation results, the proposed RDF achieved 88.9328 ~ 90.6293% precision for performance inference for a given workload and resource amount, so on the basis of the models, resources can be precisely designed in accordance with the cloud performance intent. Furthermore, RDF exceeds the conventional resource design approach in the aspects of time cost, human cost, and resource design results.

Additionally, to reduce intent breach risk and thus enhance user satisfaction in commercial cloud service delivery, we have validated the intent breach prevention mechanism (P-mode) for RDF. The proposed P-mode models P_{BDP}, P_{BEP}, and P_{BTP} are able to reduce intent breach risk significantly in the aspects of breach extent, breach duration and the number times a breach occurs at the price of a small precision trade-off. It is also observed that the specific P-mode model designed on the basis of a certain intent

breach penalty pattern naturally outperforms other P-models for the intent breach risk evaluation matrix that is based on the same intent breach pattern. Thus the administrator of RDF could choose the type of P-mode in accordance with the intent breach penalty pattern that is agreed between the cloud user and the cloud provider.

Ongoing and future work for ICSM includes: 1) to validate RDF's effectiveness in various cloud-based service scenarios including Virtual Network Function (VNF), etc.; 2) to carry out a trial experiment to verify its effectiveness in a real business environment; and 3) to expand the application of intent-driven management from the cloud domain to the end-to-end service by collaboration with intent-based management frameworks of other domains including the transport NW domain, wireless domain, etc.

Acknowledgements
Not applicable.

About the authors

Chao Wu (Research Engineer, NTT Access Network Service Systems Laboratories) received the bachelor degree in engineering from Zhejiang University in 2009 and the master degree in engineering from Waseda University in 2013. In 2014, she joined NTT Access Network Service Systems Laboratories, where she has been researching and developing management mechanisms for telecommunications, especially in cloud and virtualization.
Shingo Horiuchi (Senior Research Engineer, NTT Access Network Service Systems Laboratories) received a B.E. and M.E. in engineering from University of Tokyo, in 1999 and 2001. He joined NTT Access Network Service Systems Laboratories in 2001. He has been researching and developing access network operation systems. He has been engaged in the standardization work for operation support systems in TM Forum as a member of the Open Digital Architecture Project since 2014. He is a member of IEICE.
Kenji Murase (Senior Research Engineer, NTT Access Network Service Systems Laboratories) received a B. E and M.E. in engineering from Waseda University, Tokyo, in 2004 and 2006. He joined NTT Communications the same year and is currently engaged in developing operation support systems of access networks.
Hiroaki Kikushima (Research Engineer, NTT Access Network Service Systems Laboratories) received a B.E. and M.E. in electrical engineering from the University of Yamanashi, in 1994 and 1996. He joined NTT Software Headquarters in 1996. He also worked at NTT COMWARE's AI Business Strategy Office, where he created various new services.
Kenichi Tayama (Senior Research Engineer, Supervisor, Group Leader, NTT Access Network Service Systems Laboratories) received a B.E. and M.E. in electrical engineering from the University of Electro-Communications, Tokyo, in 1993 and 1995. He joined NTT Optical Network Systems Laboratories in 1995. He also worked at NTT EAST's IT Innovation Department and NTT-ME's Network Operation Center, where he researched and developed network operations support systems. He is a member of IEICE.

Authors' contributions
Chao Wu and Horiuchi Shingo jointly designed and directed this research. Kenji Murase, Hiroaki Kikushima contributed to the validation experiment of this work. Kenji Tayama contributed to the design of framework of this work. All authors reviewed and approved the final manuscript.

Funding
Funding information is not applicable.

Availability of data and materials
Since the data and the code used in this work are confidential information of NTT Laboratories, and will be used for further studies, we cannot provide them to the public.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 11 December 2020 Accepted: 19 March 2021
Published online: 02 June 2021

References

1. ETSI GS ENI 005 V1.1.1 Experiential Networked Intelligence (ENI) System Architecture (work in progress). https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp? WKI_ID=5408. Accessed 20 Feb 2021
2. DGR/ENI-0013 Experiential Networked Intelligence (ENI) Intent Aware Network Autonomicity" (work in progress). https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp? WKI_ID=58217. Accessed 20 Feb 2021
3. Autonomous Networks: Empowering Digital Transformation for Smart Societies and Industries. https://www.tmforum.org/resources/whitepapers/autonomous-networks-empowering-digital-transformation-for-smart-societies-and-industries/. Accessed 20 Feb 2021
4. 3GPP TR 28.812 Telecommunication management: study on scenarios for Intent-driven management services for mobile networks (work in progress). https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3553. Accessed 20 Feb 2021
5. 3GPP TR 28.312 management and orchestration intent-driven management services for mobile network" (work in progress. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3553. Accessed 20 Feb 2021
6. DGS/ZSM-009 Zero-Touch Network and Service Management (ZSM) Closed-loop automation: Enablers (work in progress). https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp? WKI_ID=58053. Accessed 20 Feb 2021
7. ONOS Intent Framework. https://wiki.onosproject.org/display/ONOS/Intent+Framework. Accessed 20 Feb 2021
8. OpenDayLight Network Intent Composition. https://wiki.opendaylight.org/view/NetworkIntentCompositionUseCase. Accessed 20 Feb 2021.
9. Zhang H et al (2017) Demo abstract: an intent solver for enabling intent-based SDN. INFOCOM, Atlanta. https://doi.org/10.1109/INFCOMW.2017.8116514.
10. Elkhatib Y et al (2018) Benchmarking the ONOS intent interfaces to ease 5G service management. GLOBECOM, Abu Dhabi. https://doi.org/10.1109/GLOCOM.2018.8648078
11. Jacobs AS et al (2019) Deploying natural language intents with Lumi. SIGCOMM, Beijing. https://doi.org/10.1145/3342280.3342315
12. Tian B et al (2019) Safely and automatically updating in network ACL configurations with intent language. SIGCOMM, Beijing. https://doi.org/10.1145/3341302.3342088
13. Callegati F et al (2017) Performance of intent-based virtualized network infrastructure management. ICC, Paris. https://doi.org/10.1109/ICC.2017.7997431
14. E. J. Scheid, et al. (2017) INSpIRE: integrated NFV-based intent refinement environment. In: IM, Lisbon. doi: https://doi.org/10.23919/INM.2017.7987279
15. Aklamanu F et al (2018) Intent-based real-time 5G cloud service provisioning. Globecom, Abu Dhabi. https://doi.org/10.1109/GLOCOMW.2018.8644457
16. Kang J et al (2017) LMS: label management service for intent-driven cloud management. IFIP/IEEE IM, Lisbon. https://doi.org/10.23919/INM.2017.7987278
17. Kim J, Kim E, Yang J, Jeong J, Kim H, Hyun S, Yang H, Oh J, Kim Y, Hares S, Dunbar L (2020) IBCS: intent-based cloud Services for Security Applications. IEEE Communication Magazine 58(4):45–51. https://doi.org/10.1109/MCOM.001.1900476
18. He L, Qian Z (2020) Intent-based resource matching strategy in cloud. J Information Sci 538:1–18. https://doi.org/10.1016/j.ins.2020.05.045
19. Kuwahara T et al (2021) An intent-based system configuration design for IT/NW services with functional and quantitative constraints. IEICE Trans Commun. https://doi.org/10.1587/transcom.2020CQP0009
20. Chung C et al (2020) A Design of IoT Device Configuration Translator for Intent-Based IoT-Cloud Services. ICACT, PyeongChang. https://doi.org/10.23919/ICACT48636.2020.9061282
21. C. Wu and H. Shingo (2018) Intent-based Service Management. In: ICIN 2018, Paris. . doi: https://doi.org/10.1109/ICIN.2018.8401600

22. de Myttenaere A, Golden B, le Grand B, Rossi F (2016) Mean absolute percentage error for regression models. Neurocomputing 192:38–48. https://doi.org/10.1016/j.neucom.2015.12.114
23. Cort J. Willmott, et al. (2005) Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. Clim Res 30(1):79–82. https://doi.org/10.3354/cr030079
24. Chai T, Draxler RR (2014) Root mean square error (RMSE) or mean absolute error (MAE)? - arguments against avoiding RMSE in the literature. Geosci Model Dev 7(3):1247–1250. https://doi.org/10.5194/gmd-7-1247-2014
25. Stress-ng. http://manpages.ubuntu.com/manpages/zesty/man1/stress-ng.. Accessed 20 Feb 2021
26. M. Abadi, et al. (2016) Tensorflow: Large-scale machine learning on heterogeneous distributed systems. https://arxiv.org/abs/1603.04467. Accessed 20 Feb 2021

Publisher's Note