Tier-Centric Resource Allocation in Multi-Tier Cloud Systems

Jyotiska Nath Khasnabish, Mohammad Firoj Mithani, *Member, IEEE*, and Shrisha Rao, *Senior Member, IEEE*

Abstract—In IT service delivery and support, the cloud paradigm has introduced the problem of resource over-provisioning through rapid automation (or orchestration) of manual IT operations. Due to the elastic nature of cloud computing, this shortcoming ends up significantly reducing the real benefit, viz., the cost-effectiveness of cloud adoption for Cloud Service Consumers (CSC). Similarly, detecting and eliminating such over-provisioning of cloud resources without affecting the quality of service (QoS) is extremely difficult for Cloud Service Providers (CSPs) since they have visibility only into the state of the IT services (cloud resources) but none into the actual performance of business services. In this paper, we propose Tier-centric Business Impact and Cost Analysis (T-BICA), a tier-centric optimal resource allocation algorithm, to address the problem of rapid provisioning of IT resources in modern enterprise cloud environments, through extensive data gathering and performance analyses of business services in a simulated environment emulating a mature cloud service provider. We have derived improved analytics to address the issues and to accelerate real cloud adoption for large enterprises within the context of meeting (or exceeding) business service level objectives (SLOs) and minimizing the cloud subscription cost (OpEx) for the business. While investigating the problem, we consider the time and the cost of delivering business service in medium- to large-size enterprise environments, quantifying the negative impact of IT resource over-provisioning (due to highly mature IT services centric orchestration capabilities) on the business, and indicate how the suggested cloud analytics could assist in reducing total cost of ownership (TCO) of the business service. From our analysis of the test data, we have observed that our suggested approach and analytic reduces the cost of delivering business services by 65.19 percent, and improves the performance (total time to deliver) by 74.18 percent when compared to the existing modern cloud management and resource provisioning approach. Using T-BICA dramatically reduces upfront costs (CapEx) for CSPs (from the capacity procurement and management points of view) through efficient on-demand resource de-provisioning, without affecting business SLOs and IT service level agreements (SLAs). The improved dynamic allocation of resources also makes for better efficiency of utilization, which in turn has desirable consequences for sustainability, and makes this an approach for "Green" IT.

Index Terms—Cloud computing, infrastructure as a service, public cloud, private cloud, business impact analysis, business service, SLO, SLA, business applications, multi-tier system, over-provisioning, cloud service provider, cloud service consumer

1 Introduction

Efficient IT resource allocation in enterprise cloud systems can help organizations improve their returns on investment (ROI) either by migrating business services to public or private clouds (IaaS) or by subscribing standardized business service functions such as software as a service (SaaS) or platform as a service (PaaS) from respective CSPs [2]. In this paper, we propose Tier Centric Business Impact and Cost Analysis (*T-BICA*), an optimal resource allocation algorithm which aims to address the issue of over-provisioning of resources caused by traditional on-demand resource allocation in multi-tier cloud systems. Modern cloud-based business services often consist of multiple tiers, e.g., to handle massive and complex real-time online transaction processing (OLTP) applications hosted by organizations. T-BICA is designed to help organizations significantly minimize the CapEx (capital expenditure) and OpEx (operating expenditure) of deploying and supporting the required hosting environment. Most of the applications hosted on cloud require on-demand resource allocation capability from the cloud service providers to deal with unexpected spikes in demand. T-BICA precisely addresses this problem, and our analyses indicate the significant benefits of using the same.

The resource allocation mechanism proposed in this paper differs from the traditional resource allocation approach currently adopted in enterprise private clouds. Although there have been several resource allocation algorithms proposed, these approaches fail to monitor resource requirement and utilization at each individual tier of business services. Some of these approaches are to use local and global allocation algorithms for providing on-demand capacities of concurrent applications [3], using a Cloud Capacity Manager (CCM) to provide on-demand compute capacity management for virtualized data centers [4], using virtualization technologies [5], [6] for allocating resources dynamically based on application demands that can combine different workloads and improve utilization of resources, and using an optimal cloud resource provisioning

- *J.N. Khasnabish is with DataWeave Software Pvt. Ltd. E-mail: jyotiskanath.khasnabish@iiitb.org.*
- *M.F. Mithani is an independent researcher and cloud evangelist based out of Melbourne, Australia. E-mail: mf.mithani@ieee.org.*
- *S. Rao is with the International Institute of Information Technology-Bangalore, Bangalore 560100, India. E-mail: shrao@ieee.org.*

Manuscript received 23 Apr. 2014; revised 24 Feb. 2015; accepted 13 Apr. 2015. Date of publication 21 Apr. 2015; date of current version 6 Sept. 2017. Recommended for acceptance by O. Rana.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TCC.2015.2424888

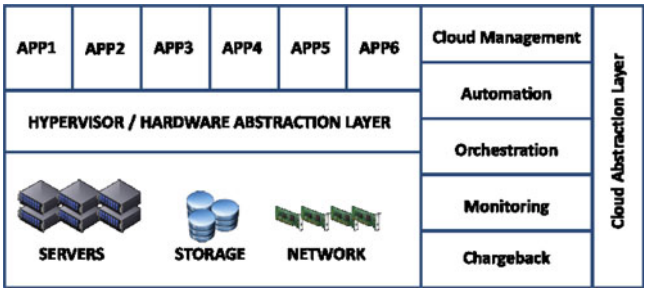

Fig. 1. Generic cloud architecture.

algorithm [7] for provisioning of compute resources for short term on-demand plans as well as long term plan. There also have been several game-theory based resource allocation algorithms proposed which try to optimize resource consumption in the cloud [8], [9], [10]; there also is an SLA-based resource allocation algorithm for SaaS providers [11]. But none of these approaches talks about provisioning resources at the tier level. A typical multi-tier business system usually consists of web, application and database tiers, which have their own utilization rates and resource demand rates. The usual gross approach that does not take tier-level resource utilizations, actual load as well as overall business service performance into account to trigger only justified additional capacity resource provisioning, is inherently limited in its performance. Our experiments show that T-BICA is able to allocate resources at tier level and utilize resources more efficiently, which leads to lower resource subscription costs and helps avoid SLA violations.

The modern cloud computing approach has well-known advantages over conventional grid computing which is purpose designed and deployed for special-purpose systems [12], [13], [14]. A wide variety of analytic (predictive and forecasting) and business tools (business impact analysis), and highly sophisticated cloud lifecycle management (CLM) suites for rapid automation and end-to-end process orchestration for IT Service Management (ITSM) processes are deployed in large organizations to harvest the benefits of cloud computing for business outcomes [2], [15]. Fig. 1 shows the generic architecture of a cloud environment. The environment consists of an application layer where several business applications can be hosted and can be co-located and managed by single CLM [16], [17]. Nevertheless, a virtualized hosting environment is highly elastic compared to a physical (server) cloud environment. The National Institute of Standards and Technology (NIST) defines cloud deployment models as being for public, private, hybrid and community clouds, on the basis of management, purpose and utility (end user), ownership and operation of the cloud environment [18]. The ideal cloud deployment model for an organization could be a combination of two or more deployment models, and the decision is highly influenced by business processes and business services, rather than the availability of cloud deployment technologies [19], [20].

Efficient IT resource allocation to business services improves ongoing savings, as the IT operation process matures, and the cloud environment graduates from basic (IT SLA-centric) to advanced (business SLO-sensitive). The currently available public and private cloud technologies and services are far more mature compared to conventional IT delivery (physical and virtualized), in terms of resource allocation and delivering business service SLOs. As we explain the current resource allocation approach followed by most CLM and CSPs, it becomes evident that the resource allocation rules designed for the systems are focused on uptime, availability and performance of end-to-end business service transactions, rather than effective utilization of individual building blocks (IT resources). This gross approach suffices to achieve business objectives, but it increases costs (OpEx) for CSCs by way of higher cloud subscriptions, and also makes for higher CapEx for CSPs (to reserve sufficient capacity). T-BICA significantly reduces both the OpEx and the CapEx, and helps achieve business SLOs at lower costs.

In the conference extended abstract [1], some of these ideas were presented briefly, but here we offer the following improvements *in extenso*:

- We have given a stronger technical foundation for the technique proposed, in the form of algorithms as well as mathematical reasoning, in Section 3. Detailed analyses of the working of the proposed method and its cost benefits are also added.
- We have drawn a clear distinction between the traditional resource allocation techniques adopted by cloud providers currently (Section 2.2), and T-BICA. While it might be expected that the academic research is ahead of the practice, so that this problem should not appear in systems proposed and discussed by researchers, our literature review shows instead that even papers in learned conferences and journals [21], [22], [23] do indeed propose provisioning (and analyses for QoS and other parameters) considering a cloud-based application (even in explicitly multi-tier systems) as a whole block, not considering loads and performances at tier levels, though this leads to gross wastage of resources [24].
- We have indicated the working of the conventional approach with a static resource allocation algorithm (Section 2.3), with a detailed example concerning a personal banking service, to demonstrate batch job execution in the cloud environment traditionally. Later, we have indicated the working of the T-BICA approach using the same parameters as the example, to illustrate the realistic advantages gained.

The rest of the paper is organized as follows. Section 2 discusses existing resource allocation algorithms for cloud systems. Sections 2.2 and 2.3 discuss in general form the resource allocation approach that is currently adopted by CSPs, and its shortcomings. Section 3.1 explains our proposed T-BICA algorithm in detail and give proofs of correctness using lemmas and a theorem. In Section 4, we perform experiments and data analysis (for a simulated environment) on a industry-grade three-tier savings bank account management business service to show the shortcomings of the traditional cloud resource allocation approach of resource allocation in private cloud and how T-BICA helps in improving the resource allocation at justified cost.

2 Resource Allocation in Cloud Systems

2.1 Related Work

There have been several proposed algorithms for resource allocation in cloud systems. Chang et al. [25] present an

polynomial-time approximation algorithm for the resource allocation problem, which is formulated according to the demand for computing power and other resources. That solution can be exploited by business organizations to reduce cost and overheads associated with cloud management. An et al. [26] present a negotiation-based resource allocation scheme where providers and consumers automatically negotiate resource leasing contracts. This scheme also allows agents to breach and de-commit their contract at pre-specified penalty. The paper compares several existing resource allocation algorithm and shows that the proposed approach achieves higher social welfare. Rai et al. [27] present a generic and extensible tool Wrasse which tries to solve the problem of resource allocation in cloud. It is powered by GPUs in the back-end which makes it possible to provide solutions to resource allocation problem quickly with minimum latency. A two-tiered resource allocation mechanism [3] has been proposed by Song et al. for providing on-demand capacities of concurrent applications using local and global resource allocation algorithms. Cloud Capacity Manager [4] is a prototype system which provides an on-demand compute capacity management for virtualized data centers consisting of thousands of machines. Also, Chaisiri et al. [7] propose an optimal resource provisioning approach for compute resources, given short-term on-demand plans as well as long-term plans. Analytic Hierarchy Process [28] is a complex decision-making technique which decomposes a problem into multiple sub-problems to reach a final decision. A task-oriented resource allocation technique based on AHP has been proposed for a cloud computing environment [29]. Han et al. [30] present an adaptive elastic scaling algorithm which reduces the cloud infrastructure cost to allow scaling applications at bottleneck tiers. Goudarzi and Pedram [31] discuss an SLA-based resource allocation problem for multi-tier applications using cloud computing.

With the increasing maturity of virtualization platforms and availability of high-capacity compute resources, the cloud has become the preferred option for organizations to consider moving non-critical business services, rather than expanding existing data centers or investing in new ones. Therefore, organizations are increasingly migrating seasonal workloads to public clouds through the hybrid cloud model and retaining critical business services in private clouds [32], [33], [34]. This approach enables organizations to reduce the TCO of their business services, and makes them more competitive in the market. Eventually, as cloud product offerings get mature, and as the cloud operations processes gets more efficient, the unit cost of cloud resources tends to decrease year after year. Such reduced costs for cloud resources encourages rapid cloud adoption, and some of the credit for this goes to economies of scale in public cloud environments [35]. However, wasteful allocation of resources tends to offset the advantages of reduced costs [36], [37].

There have been some parallel data processing platforms like Nephele [38] which exploits the dynamic resource allocation for IaaS-based applications which process massive amounts of data. Nephele is the first data-processing framework to use dynamic resource allocation on cloud for both task scheduling and execution. Yazir et al. [39] present a

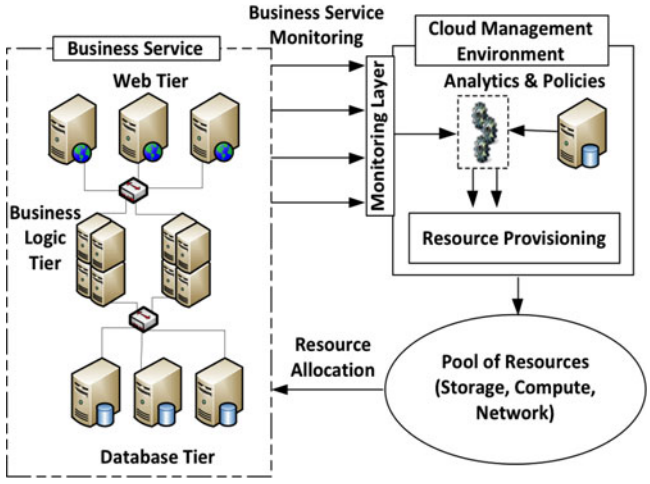

Fig. 2. Standard business service monitoring and resource allocation in private cloud.

Multiple Criteria Decision based dynamic resource allocation policy on cloud using the PROMETHEE method [40]. In their architecture, resource management is decomposed into an array of independent tasks, each of which is carried out by Autonomous Node Agents. The simulations show the scalability and flexibility of the distributed architecture.

All these approaches fail to observe tier-level resource utilization and allocation. A white paper by Cloudyn [24] says that over-provisioning of cloud resources to handle all contingencies has become an "epidemic." We observe that for many years, starting from the earliest models of web services [41], to more recent works dealing with multi-tier applications and services [42], [43], and even more recent work applicable to cloud environments in multi-tier systems [21], [22], [23], the tendency has been to consider the system as a whole, ignoring its internal tiered structure, and to analyze it accordingly. Thus, though various important ideas and algorithms are applied to improve QoS and other attributes, performance analyses and resource allocation are considered for the whole system at once, not in a tier-wise manner.

T-BICA monitors resource utilization at individual tiers, ensuring the business service SLOs are within agreed ranges. In the following sections, we show how not considering tier-level resource allocation is a shortcoming, and how T-BICA can replace the traditional resource allocation approach by monitoring and allocating resources on-demand at each tier of the business service.

2.2 Shortcomings of the Current Approach

We consider a scaled-down version of an enterprise class business service (Fig. 2) of the finance industry to demonstrate the shortcomings of the current elastic cloud model and on-demand cloud provisioning, and the unjustified overhead of associated cost incurred by CSCs due to the same.

The application used here is a variant of the widely-used personal banking system that maintains individual savings bank accounts. This is a three-tier (web, application and database) business service that generates bank customer account statements on a monthly basis after calculating total interest earned for the month, and sends account statements

to the respective account holders. All the business services and functions of the banking, financial service and insurance (BFSI) are highly regulated by the central banking authorities at national level, and other controllers. Therefore in this case it is binding on the bank to send monthly account statements to account holders with up-to-date account balance and interest earned. Any delay beyond a certain limit may cost the bank a heavy penalty or even loss of license. However, for the sake of simplicity, we assume that the bank must complete the entire process of updating individual account balance, generating account statements within the first seven days of each month (or quarter), and sending each statement by e-mail (or snail mail) to the respective account holder. Our simulation is limited to generating a large number of account statement and saving them into a database, but not actually generating e-mails/snail mails—generation and delivery of such notifications typically belongs to a different department, and requires a different system, that is not in the scope of our work. (There is presumably little research interest in that function, as well.) Therefore, we do not consider the processing load of sending the specified number of statements to the associated account holders, which is a constant regardless of what methods and resources are used to generate them. Fig. 2 shows the resource allocation process in a private cloud environment with multiple tiers involved, and also illustrates how business services are monitored at the Monitoring Layer.

However, the conventional resource allocation approach has the following shortcomings.

1) It does not monitor precise resource utilization at individual tiers of the business service; it only monitors business SLOs.
2) It does not do effective on-demand resource allocation across tiers, because it uses static orchestration rules and considering overall business service performance but not tier-level performance.
3) It considers criticality of business service to complete the job, but not the relative loads on the tiers.
4) It tends to magnify the negative impact on business SLOs considering higher costs, longer times to complete, and loss of credibility (goodwill) of the organization (due to delay in completing jobs because of degraded performance of one or more tier(s) of the business service).

An efficient and effective approach of IT resource allocation needs to be followed to achieve and maintain QoS of business services [44]. Every tier of a business service has unique IT resource requirements, and dissimilar load profiles. For instance, the business logic tier (application tier) in the case of personal banking system may have massive load spikes while executing monthly batch jobs, compared to a more uniform load on the web tier (presentation tier), while the database tier may have moderate load, which increases only while changes are committed. Due to disparate workload profiles, each tier requires unique consideration while assigning resources, to ensure that the business service meets the SLOs without wasteful IT resource usage.

As illustrated in Fig. 2, the resource utilization and processing overhead are monitored through the monitoring layer by the CLM. The default policies and custom resource provisioning rules trigger the provisioning of additional resources whenever the overall business service performance (SLO) drops below some pre-set threshold. In the conventional approach, the CLM triggers additional resource provisioning across all tiers of the business service to ensure the business service delivers as expected by the business. Since the individual tier performance and resource utilization are not monitored by the CLM, the associated provisioning rules add resources across tiers from an unutilized pool of resources (until this is exhausted), to meet SLAs.

2.3 Traditional Resource Allocation Algorithm

The type of enterprise class business service we have considered for the data analysis and to validate the T-BICA approach is considered a highly critical business service in the BFSI (banks, financial services, and insurance) industry. Therefore, the workload (business service) cannot be hosted in public cloud environments (e.g., Amazon Web Services, Microsoft Azure) considering personally identifiable information (PII) and other information security and data protection compliance requirements of the industry, which is governed by laws such as the Sarbanes-Oxley Act (SOX), Basel II, and relevant provisions of the U.S. Patriot Act. Similarly, tier-level resource allocation is already implemented in public cloud environments using auto-scaling algorithms (e.g., AWS and Azure) to provision resources automatically as the resource utilization for respective tiers (i.e., virtual machines of some tier) go beyond certain levels. However, the public cloud auto-scaling and on-demand resource provisioning does not consider overall impact on business when the performance (i.e., SLOs) of the business service degrades. For example, even if all the tiers of the business service are sufficiently provisioned, the business service might not be able meet the performance criteria set by CSCs (performance, response time, etc.). Therefore, mere on-demand provisioning of resources (VMs, storage, network) to keep all tiers of the business services well supplied does not guarantee that the service meets desired standards and has no adverse impact on the business. Likewise, a retail personal banking system cannot be hosted on a public cloud due to data security laws, as well as the intrinsic approach of public cloud providers of provisioning resources and delivering only IT services rather than business services, monitoring IT SLAs (resource utilization at VM level) rather than monitoring the performance of the whole business service, etc.

A major challenge with existing private cloud environments in large organizations is that they have two distinct types of monitoring tools (layers) in place for their enterprise workloads, which do not work collaboratively without active human intervention for decisions involving both. One monitors IT services utilization (i.e., of vCPU/Core, RAM, Storage, Network) to ensure that the hosted business applications are kept supplied by triggering on-demand auto-provisioning of resources if the loads on the VMs go beyond pre-set thresholds. The second type of tool, called business performance management (BPM), checks the overall

TABLE 1
Notation

Notation	Description
w, a, d	Tiers in a multi-tier business service – web, application, database
η_w, η_a, η_d	Allocated resource to each of the tiers
x, y, z	Incremental allocation to each tier, x server(s) to web tier, y unit resources to application tier, z unit resources to database tier
$\zeta_w, \zeta_a, \zeta_d$	Unit amount of resources for each tier
$\upsilon_w, \upsilon_a, \upsilon_d$	Current workload on each tier
τ_{bs}	Business service threshold in traditional resource allocation
τ_w, τ_a, τ_d	Threshold for each tier of the business service in T-BICA
τ_{min}	Minimum tier threshold parameter for T-BICA
$\iota_w, \iota_a, \iota_d$	Minimum number of resources each tier must have to meet SLO
$\kappa_w, \kappa_a, \kappa_d$	Maximum number of resources each tier can have
`allocate()`	Function which starts the allocation of resources in the traditional approach
`tbica_allocate()`	Function which starts the allocation of resources in T-BICA
ρ	Represents the capacity of resources stored in cloud resource pool
BC	Represents the initial Basic Configuration state of the cloud system
$BC + i$	Represents the ith state after the initial configuration

performance by monitoring across the tiers (web, application, database) of the business service.

The BPM monitors the overall performance of workloads, and as it detects slow performance (by monitoring time to complete each transaction, end-to-end), it triggers an "event based alert" to activate on-demand provisioning of resources that is facilitated by the CLM (orchestration layer) to provision unit quantity of resources across tiers to improve performance of the workload in a matter of minutes.

In Table 1, we indicate the mathematical notation and terminology used in the algorithms and elsewhere.

For calculating the TCO of the system, there are fixed costs, along with variable costs, for each tier t.

Algorithm 1. Static Resource Allocation Algorithm

1 $\eta_w \leftarrow x \times \zeta_w$;
2 $\eta_a \leftarrow y \times \zeta_a$;
3 $\eta_d \leftarrow z \times \zeta_d$;
4 **while** *job is running* **do**
5 **if** $\{\upsilon_w > \tau_{bs} \,||\, \upsilon_a > \tau_{bs} \,||\, \upsilon_d > \tau_{bs}\}$ **then**
6 **if** $\{\eta_w + \zeta_w \leq \kappa_w \,\&\, \eta_a + \zeta_a \leq \kappa_a \,\&\, \eta_d + \zeta_d \leq \kappa_d\}$ **then**
7 $\{\eta_w, \eta_a, \eta_d\} \leftarrow allocate(w, a, d)$
8 **end**
9 **end**
10 generate_statement();
11 **end**
12 **if** *job successfully finished* **then**
13 **foreach** *tier* t, $t \in \{w, a, d\}$**do**
14 deallocate additionally allocated resources;
15 **end**
16 **end**

Algorithm 1 summarizes the traditional resource allocation process followed in cloud environments. We consider a personal banking business service to demonstrate batch job execution in the cloud environment. As the load on the system increases, the monitoring layer updates the CLM and additional resources are provisioned across tiers to ensure the business service meets the agreed-to SLAs until all available resources from the reserve pool are fully utilized by the service. The TCO of business service comprises fixed cost (CapEx) as well as variable cost (OpEx) for each tier t until the job is completed.

Algorithm 2. Static Resource Allocation Function

1 **Function** `allocate`*(w, a, d)*
2 **if** $\{\zeta_w + \zeta_a + \zeta_d \leq \rho\}$ **then**
3 $\rho \leftarrow \rho - \{\zeta_w + \zeta_a + \zeta_d\}$;
4 $\eta_w \leftarrow \eta_w + \zeta_w$;
5 $\eta_a \leftarrow \eta_a + \zeta_a$;
6 $\eta_d \leftarrow \eta_d + \zeta_d$;
7 **end**
8 **return**$\{\eta_w, \eta_a, \eta_d\}$;

Algorithm 1 depicts stages of the business service throughout the processing of the job (generation of personal savings account statements), until it completes the jobs by generating a pre-decided number of statements within a specific time-frame. The algorithm starts in the no-load state where the batch process is not yet initiated by the business logic (application tier), and overall load on the system is low.

The business service environment has some minimum number of unit resources assigned to each tier when there is no load on the system. We denote the initial allocation of unit resources per tier as $\{x, y, z\}$ which is defined as $\{\eta_w, \eta_a, \eta_d\}$. From line 4 until 11, the system goes through massive workload processing stage while processing the batch jobs (traversing through loops) until all the jobs are processed (all the statements are generated) by the business service. The **if** condition indicates that if any of the tiers has a processing overhead more than a maximum threshold set, a specific amount of IT resources (compute, storage, network) are allocated to the business service through on-demand resource provisioning by the CLM. The allocations of the resources are facilitated by the `allocate()` method (as depicted in Algorithm 2). The method validates the available capacity in the resource pool by calculating the total amount of resources required to provision across the tiers of the business service in one iteration of resource provisioning. The CLM provisions unit resources across tiers,

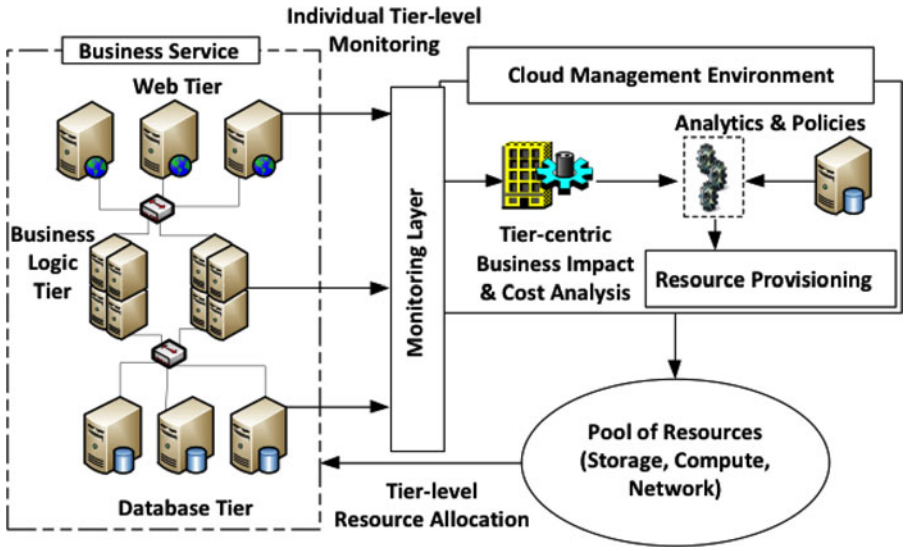

Fig. 3. Tier-centric business impact and cost analysis in private cloud.

which reduces the total available capacity of the free resource pool by the total number of units provisioned. However, it does not look at the current workload on individual tiers while provisioning the resources from the pool, resulting in under-utilized tiers also being provisioned some resources, along with tiers that are heavy utilized. This causes over-provisioning and wastage of resources, and also tends to cause the resource pool to run out faster. The business service health and SLO compliance are monitored by the CLM with business analytics to ensure it has enough resources to process the batch jobs. The CLM provisions additional resources from the pool in successive iterations each time the business service SLO breaches some set maximum threshold, and goes on provisioning until the SLO is within the threshold, or until all the available capacity is exhausted (lines 1–8, Algorithm 2). The business service performance and time to deliver the business results suffer when it runs out of available capacity to provision additional resources as the business service hits the maximum threshold τ_{bs}. In such situations, the performance of the business service degrades until additional resources are procured at premium cost. The degraded business service executes jobs until all are completed successfully. Due to over-provisioning and ineffective resource utilization, the batch job processing may also go on for an extended period of time, breaching business SLOs and negatively impacting end user experience of using the banking services. When all jobs are completed successfully, the CLM starts de-provisioning all additionally allocated resources (while retaining a minimum number units of the resources per tier) back to the resource pool to use for other business service(s).

When we compare this traditional approach with other resource allocation problems, such as fair division and cake-cutting [45], [46], we can understand its limitations. Considering each tier to be a distinct individual seeking resources, static allocation is not Pareto efficient [47], because it may be possible to improve outcomes for some tier without making any other tier worse off.

Remark 2.1. Any static allocation approach, e.g., allocating resources to tiers in fixed proportions though not in identical quantities, is not Pareto optimal, and is hence subject to Pareto improvement.

This is so, because in any static allocation approach, the allocation of resources to various tiers is liable to be out of sync with the actual utilizations of the tiers, thus making some tier starve while some other has allocation in excess of its needs.

The way around this is obviously to monitor each tier's allocation *vis-à-vis* its needs, and make changes accordingly. This is dealt with next.

3 Improved Approach and Solution

The overall business service SLOs are limited by the performance, uptime, and availability of the "weakest link" of all the tiers of the business service. Therefore, monitoring each tier of the business service ensures that the business service meets (or exceeds) the agreed SLO (to deliver business values and outcomes). Better performance and response time improves the end user experience of using the service(s), and secondarily helps improve the reputation and market share of the organization by delivering high quality services. From a business point of view, delivering quality services at an attractive price is critical for the success and survival of the business function and services.

3.1 Tier-Centric Business Impact and Cost Analysis

The T-BICA approach (see Fig. 3) ensures that the required quantity of resources are allocated at each individual tier of the business service by considering its processing overhead, performance, uptime and availability to ensure that the business SLO is met. T-BICA also de-provisions resources from tiers that are underutilized to repurpose these resources for overloaded tier(s) of the business service. Overall, T-BICA leverages existing monitoring, CLM (analytics and policies) to provision resources efficiently and assist the CLM to utilize the available resource pool to improve business service performance at justified cost. Fig. 3 indicates the role of T-BICA in the business service.

We place T-BICA between the monitoring layer and the analytics and policy-implementation system, to ensure all data gathered by the monitoring layer are analyzed by T-BICA before the analytics can deploy appropriate resource provisioning policies. The tier-level data are passed through the monitoring layer to T-BICA for analysis.

3.2 T-BICA Algorithm

The T-BICA algorithm (Algorithm 3) considers additional attributes and variables to monitor individual tier performance, and tier workload thresholds (τ_w, τ_a, τ_d) for the business service (notation used is given in Table 1).

Algorithm 3. T-BICA Algorithm

```
1  η_w ← x × ζ_w;
2  η_a ← y × ζ_a;
3  η_d ← z × ζ_d;
4  while job is running do
5    if {υ_w > τ_w || υ_a > τ_a || υ_d > τ_d} then
6      foreach tier t, t ∈ {w, a, d} do
7        if {υ_t > τ_t} then
8          tbica_allocate(t);
9        end
10     end
11   end
12 end
```

The algorithm starts with no load, i.e., an idle state on the business service with minimum allocated resources (servers) per tier until the business logic (application tier) triggers the batch process and jobs are processed by the business service. As the workload increases, the business logic and database tiers have increasing processing overhead while the web tier has minimum load throughout the batch process until all the jobs are completed successfully. In conventional static orchestrated resource provisioning environments (private clouds), a CLM monitors the increase in load of the entire business service, and starts provisioning additional resources across all tiers to keep business service SLOs under control. Unlike this traditional approach, T-BICA monitors the performance and resource utilizations of each tier through the monitoring layer, and activates the CLM to allocate additional resources in successive iterations (lines 4-12) to the tier (s) of the business that experience massive loads and crossing of their thresholds (by calling `tbica_allocate()` method, Algorithm 4). T-BICA ensures tiers are allocated additional resources to maintain the business service performance and loads below maximum threshold at each individual tier until all jobs are completed. When the resource pool is completely exhausted (without enough resources to allocate to any tier), T-BICA considers individual tier resource utilizations of all tiers of the business service (from line 6, Algorithm 4), and starts de-provisioning of resources from tiers that are underutilized. The free resources are then committed back to the free resource pool and then later provisioned to the tier(s) experiencing high workload and crossing individual tier maximum threshold. While de-provisioning the resources from an under-utilized tier, T-BICA ensures that the tier retains minimum capacity so that it functions normally and stays below the preset maximum processing load threshold.

Algorithm 4. T-BICA Allocation Function

```
1  Function tbica_allocate(t)
2    if {ζ_t ≤ ρ  η_t + ζ_t ≤ κ_t} then
3      ρ ← ρ − ζ_t;
4      η_t ← η_t + ζ_t;
5    end
6    else
7      foreach tier t', t' ∈ {w, a, d}  t' ≠ t do
8        if {υ_t' < τ_t'  η_t' > ι_t'} then
9          η_t' ← η_t' − ζ_t';
10         ρ ← ρ + ζ_t';
11         ρ ← ρ − ζ_t;
12         η_t ← η_t + ζ_t;
13         break;
14       end
15     end
16   end
17   return
```

Consider the following to understand the unique properties of the T-BICA algorithm.

Remark 3.1. For a tier t in a multi-tier business service, Algorithm 3 & Algorithm 4 allocate resources if $\rho \geq \zeta_t$ and $\eta_t + \zeta_t \leq \kappa_t$.

Algorithm 3 checks all the tiers for their current workloads and compares these with their threshold limits. It tries to allocate additional resources for the tier only if the workload has exceeded the threshold limit. In that case, tier t is passed to the `tbica_allocate()` function. This assumes that the following conditions are satisfied by the current scenario:

$$\rho \geq \zeta_t, \tag{1}$$

$$\eta_t + \zeta_t \leq \kappa_t. \tag{2}$$

This in turn means the pool has enough resources to satisfy the need for the tier t given (1). Also, provisioning additional resources for tier t does not breach the limit of the per-tier resource SLO given (2). Then the `tbica_allocate()` function provisions a resource unit for tier t.

We can now see how the T-BICA allocation approach works even when (1) is not satisfied.

Lemma 3.2. *For a tier t in a multi-tier business service, where $\rho < \zeta_t$, Algorithm 3 is still able to allocate a resource if there exists a tier t', $t' \neq t$ where $\eta_{t'} > \iota_{t'}$ and $\upsilon_{t'} < \tau_{t'}$.*

Proof. We show that T-BICA is able to allocate resources when Condition 1 does not hold. We have seen that given a pool capacity greater than the unit resource requirement of tier t, T-BICA allocates extra resources for that tier. Now we consider the condition when the current pool capacity is insufficient to meet the requirement of tier t.

In Algorithm 3, tier t is passed to the `tbica_allocate()` function when there is a need to provision an additional resource for t. First we check whether we have resources in the pool to allocate, but for this case the **if** condition fails at line 2. Then we look at some other tier t' in our business service. If the following conditions

TABLE 2
Cost of Cloud Resources Per Day (OpEx)

Tier	Management and Support	CPU (cores)	RAM (GB)	Storage (GB)	Network (Gbps)
Web	$121.94	$1.44	$0.38	$0.003	$4.56
Application	$146.22	$1.44	$0.38	$0.003	$4.56
Database	$194.71	$1.44	$0.38	$0.003	$4.56

TABLE 3
BC State of Hardware and Load Thresholds

Tiers	BC State	CPU (cores)	RAM (GB)	Storage (GB)	Network (Gbps)	Default Load	Maximum Threshold
Web	3	12	64	500	3	13%	85%
App	3	24	128	800	3	17%	80%
DB	2	48	256	1,000	4	14%	75%

are satisfied, `tbica_allocate()` function provisions a unit resource for tier t

$$t \in \{w, a, d\}, t \neq t', \tag{3}$$

$$\upsilon_{t'} < \tau_{t'}, \eta_{t'} > \iota_{t'}. \tag{4}$$

If (3) and (4) are satisfied, Algorithm 3 allocates one unit resource for tier t even if (1) has not been satisfied. □

Lemma 3.2 brings us to Theorem 3.3, where we show that if the above conditions are not satisfied, it is not possible to allocate resources for any tier of the business service.

Theorem 3.3. *For tier t in a multi-tier business service, Algorithm 3 is only able to allocate resource for the tier if $\rho \geq \zeta_t$ or $\rho + \zeta_{t'} \geq \zeta_t$ for a tier tier t', $t' \neq t$ where $\eta_{t'} > \iota_{t'}$ and $\upsilon_{t'} < \tau_{t'}$.*

Proof. We start by assuming that Algorithm 3 has allocated one unit resource for the tier t and none of the conditions are satisfied. We look at (1) and try to analyze whether Algorithm 3 provisioned one unit resource from the pool or not. In order to allocate one unit resource from the pool, the strength of the pool must be at least equal to one unit server of tier t. But this scenario would eventually satisfy (1) which is not true. So, the additional unit resource could not have been assigned from the resource pool.

Now we investigate whether the unit resource was de-provisioned from some other tier and given to the tier t. In order to do that, there must have been one tier other than tier t for which the workload was below the minimum threshold and current allocation is greater than the minimum allocation limit. But that would satisfy (3) and (4) which is not true. This means the unit resource was not de-provisioned from any other existing tier and assigned to the tier t.

Remark 3.1 and Lemma 3.2 show that the only way T-BICA can allocate a unit resource for a tier t by satisfying either (1) or (3) and 4. This contradicts our initial assumption that Algorithm 3 was able to allocate one unit resource for tier t without satisfying any of the given conditions. QED. □

4 Data Analyses & Results

Environment. In this section, we describe experiments and data analyses pertaining to a personal banking business service in a private cloud environment to demystify the shortcomings of the traditional approach of resource allocation in a private cloud, and show how T-BICA helps improve resource allocation for business service at justified cost.

For the experimentation and data analysis, we have taken a scaled-down version of an industry-grade three-tier savings (retail) bank account management business service. The experimental setup was hosted on a public cloud (AWS EC2) with strict limits on the capacity of IT resources (compute, storage and network). (Though an enterprise personal banking application would not be hosted on a public cloud in practice, we use one here for a standard cost baseline, considering the variability and confidentiality of private cloud costs. It is easy to see that T-BICA is not dependent on particular cloud pricing structures, and will produce similar results given different cost parameters for a private cloud.)

The cost per unit resource is calculated considering current market rates [48] offered by the CSP for the pool of resources (Table 2). The management (efforts and tool), and support costs are derived from the premium enterprise category of support pricing of the cloud provider.

The business service has three tiers—web (presentation), application (business logic) and database. We have an initial state of the system with bare minimum resources to host the service functions called base configuration (*BC*) for each tier. BC is a state when there is no processing overhead over the service's idle state.

Cloud resources. Table 3 shows the state *BC* of the business service with individual server counts for each tier, along with total compute (CPU, RAM), storage, and network allocated. The table shows initial default workload at each tier along with maximum load threshold. The *BC* state also shows the minimum number of resources (servers) for each tier required when there is no load, i.e., the idle state. For our experiments, the free pool total capacity is shown in Table 4.

The units of the respective resources in the environment as well as in free pool are captured in discrete quantities (e.g., CPU cores, RAM in GBs, secondary storage in GBs, and network bandwidth in Gbps) to quantify the cost of each unit of resource procured (subscribed) from the service provider.

Workload. To generate a significant amount of data for the analysis, we have simulated the generation of 10 million savings account statements using the business service, which are generated synthetically to compute the actual load in the environment for the data analysis. In the *BC* configuration, the business service takes 0.23 seconds to

TABLE 4
Subscribed Resource Pool Capacity

CPU cores	RAM	Storage	Network
28	448 GB	2,300 GB	10 Gbps

TABLE 5
Pre and Post Load for Traditional Allocation

State	Web	Load Pre	Load Post	App	Load Pre	Load Post	DB	Load Pre	Load Post
BC	3	13%	13%	3	17%	17%	2	14%	14%
BC + 1	3 + 1	35%	29%	3 + 1	89%	45%	2 + 1	55%	49%
BC + 2	3 + 1 + 1	29%	22%	3 + 1 + 1	86%	33%	2 + 1 + 1	49%	44%
BC + 3	3 + 1 + 1	22%	22%	3 + 1 + 1	82%	82%	2 + 1 + 1	44%	44%

TABLE 6
Unutilized Resource in the Pool at Each State

State	CPU Left		RAM Left		Storage Left		Network Left	
BC	84	100%	448	100%	2,300	100%	10	100%
BC + 1	48	57%	257	57%	1,368	59%	7	70%
BC + 2	18	21%	66	15%	436	19%	4	40%

generate one account statement, and therefore 26.62 days to generate the 10 million account statements.

Constraints. As per standard BFSI regulations set by central banks or similar fiscal authorities worldwide, all account statements must be delivered to their respective account holders within the first week of each quarter. Therefore, we assume that our system must observe the regulations and should deliver all the statements in less than seven days. The expectations can only be met if the business service performance is improved and it remains available 99.9 percent (or higher) to complete the jobs. The statement generation is a batch process triggered by the business logic (application tier) of the business service. The business logic extracts the account data from the database tier and calculates applicable interest on the minimum monthly/quarterly balance of each savings account and computes the total balance of the account. The final results are committed back to each account record in the database. Therefore, most of the processing is done at the application tier while the database tier provides all the data (read) and commits the updates (write) as the job is completed. Along with this, the application tier generates individual account statements and forwards them to an external dispatch system which sends them to the respective account holders by e-mail or snail mail, using contact information from their respective accounts (read from the database tier). Throughout the process, the web tier is mostly idle but experiences moderate workload if any of the account statements are manually requested through the web tier (by end users online, or bank employees at bank/branch counters).

4.1 Traditional Resource Allocation Approach

In the traditional static resource allocation approach, new resources are provisioned across tiers of the service when one (or more) tier(s) is overloaded, as we have indicated in Algorithm 1. This resource provisioning is performed by the CLM and the data is given by the cloud monitoring layer to the management layer. Once the CLM has been provided with the current lower performance matrix through the monitoring layer of the business service (due to overloaded situation of one or more tiers, it triggers the on-demand resource allocation process and starts allocating unit resource to each tier in successive increments until business service performance reaches acceptable levels, or until the CLM consumes all resources from the free pool. Once the allocation is complete, we call this a new state, transitioned from the previous state. This chain of actions goes on until the jobs are completed.

In Table 5, we show the load on each individual tier before provisioning and after provisioning the resources from the subscribed free resource pool. In Table 6, we show the unutilized resources available after each state in the resource pool in terms of CPU, RAM, Storage and Network. In both these tables, we have shown data for all three tiers—web, application and database.

Now, we demonstrate how resources are provisioned at each state starting from *BC*, using the traditional static resource allocation approach.

4.1.1 BC State

This is the initial state where no extra resources have been provisioned by the CLM yet. The business service runs in this state from the beginning and continues until significant processing load is put on the business service. From Table 5, we can see that the loads on all tiers are well below their threshold limits. Therefore no additional resources are provisioned. In Table 6, the available unutilized resources are also same in the *BC* state. In the next state, i.e., $BC + 1$, we observe that the load on the application tier has gone up to 89 percent which is over the threshold for the tier, while for the web and database tiers the resource utilization (processing load) is well below their respective thresholds.

4.1.2 BC + 1 State

At the end of *BC* state, the load on the application tier has crossed the maximum threshold. As a result, the cloud management layer starts orchestrating resource provisioning for all the three tiers. Unit resources $\{\zeta_w, \zeta_a, \zeta_d\}$ are added to all the tiers from the pool. Table 5 shows that the load on the application tier has gone below the threshold. The current strength of resource pool after the allocation is shown in Table 6. At the end of $BC + 1$ state, Table 5 shows that the load on the application tier has gone up to 86 percent which leads to state $BC + 2$.

4.1.3 BC + 2 State

As the load on the application tier crosses the threshold at the end of $BC + 1$, the CLM triggers the resource provisioning process, which brings us to the $BC + 2$ state. In the $BC + 2$ state, a unit resource for each tier is added from the pool. The current capacity available in the pool at the end of $BC + 2$ state is: $\{CPU, RAM, Storage, Network\} = \{18, 66, 436, 4\}$.

4.1.4 BC + 3 State

From Table 5, we see that at the end of $BC + 2$ state, the load on the application tier goes up to 82 percent and crossed the threshold. But Table 6 shows that we need 82 more CPU units to provision additional unit resources for all tiers, which we do not have. So in $BC + 3$ state, we are unable to provision any further resource from the resource pool, and the business service is forced to continue with the currently allocated resources to generate the rest of the account

TABLE 7
Time in Days and Cost in USD at Each State (Traditional)

State	Days	Cost
BC	26.62	$37,031.93
BC + 1	23.69	$22,537.89
BC + 2	19.66	$21,329.33
BC + 3	19.66	$21,329.33

statements. Thus, from the *BC + 3* state and onwards, the provisioned resources for all the tiers are the same as in the *BC + 2* state.

Table 7 shows the time and cost associated with the job being processed at each state from *BC* to *BC + 3*. We see that if we were to continue with the *BC* configuration, it would have taken over 26 days to generate all the statements, in clear of violation of the seven-day limit. After we have allocated additional resources from the resource pool, at the *BC + 3* state the duration reduced to 19.66 days, which still exceeds the limit. Also, the total cost for the job reduced only 42 percent from *BC* state to *BC + 3* state. It is evident that this resource provisioning policy is unacceptable and should not be adopted because of insignificant savings in operational costs, and the likelihood of regulatory problems, besides rifts with CSCs, on account of failure to meet SLOs.

Fig. 4 graphically shows the reduction in days and cost at each state. First set of points is for the initial configuration *BC*. In this state, the cost is highest which is $37,031.93 and the number of days to be taken to generate all the statements is 26.62 days. As we go through each state, the cost comes down as well as the time duration. At the *BC + 3* state, the cost remains constant—$21,329.33, as we are unable to provision more resources from the pool. Also, the time taken to complete the jobs comes down to 19.66 days and there is no further change for the subsequent states.

In Fig. 5, we show the savings in cost and number of days for every state. At the *BC* state, there is no change or no saving in either cost or duration, so the value is 0. In *BC + 1* state, we have the maximum savings that is close to $14,493.74, and the number of days reduced by 2.93. At *BC + 2* state, the savings in terms of number of days is maximum which is 4.03, and cost saving is $1,208.56. But for the subsequent states, there is no change in savings, due to the exhausted resource pool.

Here we have analyzed the scenario of bank statement generation problem with traditional resource allocation approach. We have shown that due to over-provisioning of the resources at each state accross tiers (irrespective of individual tier resource utilization and processing load), the pool gets exhausted very quickly. Hence, it becomes impossible to complete the work within the seven-day limit. In the next section, we analyze the T-BICA approach of resource allocation in details using the same example we have used in this section and demonstrate how T-BICA is able to complete the work within seven-days by monitoring workloads at each tier and allocating resources only to overloaded tier(s).

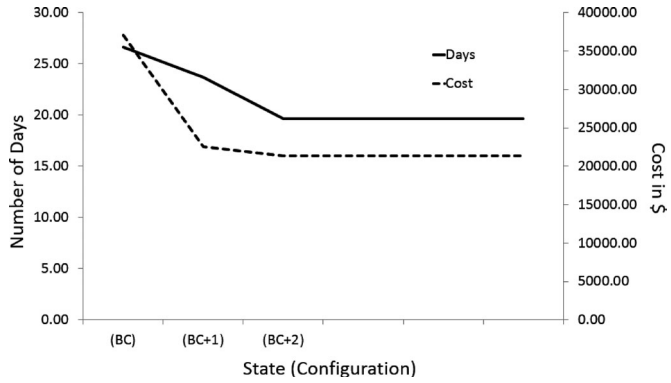

Fig. 4. Reduction in days and cost of the task (traditional).

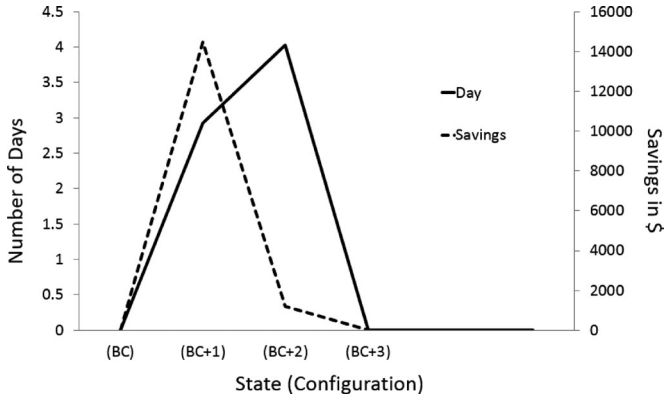

Fig. 5. Cost savings and reduction in days (traditional).

4.2 T-BICA Resource Allocation

In our proposed T-BICA approach shown in Algorithm 3, the cloud management layer monitors the load on each tier and individual tier-level performance. This results in ability to allocate additional resources at individual tiers replacing the traditional uniform resource allocation policy but ensuring that business service SLOs are monitored. The CLM function is similar to the traditional approach but the difference is in the resource allocation algorithm itself. The cloud monitoring layer continues to check on each tier's as well as whole business service performance and sends the data back to the CLM layer. When the load on a particular tier exceeds its maximum threshold, only that tier gets additional resources allocated from the resource pool.

Table 8 shows the pre- and post-resource provisioning loads starting from state *BC*. *BC* represents the initial configuration of the cloud system without any added hardware from the resource pool. States *BC + 1* to *BC + 8* show the states where extra resources have been added to only the application tier instead of uniform allocation on all tiers. This is due to the fact that only the workload on the application tier exceeded the threshold limit. At the end of *BC + 8* state, Table 9 shows that we have exhausted the resource pool. So, T-BICA de-provisions a unit resource from the web tier maintaining the minimum resource count for the respective tier, and assigns the de-provisioned resource to the application tier that ultimately brings down the load at the end of *BC + 8-1* state. Then at *BC + 9-1* state, we see that we cannot de-provision any more resources from the web tier and the database tier as they have already reached their minimum unit resource limits. So the system continues to run in its current load and finishes generating the rest of the statements. We saw in the example of traditional resource allocation approach that we run out of free resources at the end of *BC +*

TABLE 8
Pre and Post Load for T-BICA Allocation

State	Web	Load Pre	Load Post	App	Load Pre	Load Post	DB	Load Pre	Load Post
BC	3	13%	13%	3	17%	17%	2	14%	14%
BC + 1	3	13%	13%	3 + 1	89%	45%	2	55%	55%
BC + 2	3	13%	13%	3 + 2	86%	40%	2	55%	55%
BC + 3	3	13%	13%	3 + 3	81%	54%	2	55%	55%
BC + 4	3	13%	13%	3 + 4	92%	39%	2	55%	55%
BC + 5	3	13%	13%	3 + 5	87%	55%	2	55%	55%
BC + 6	3	13%	13%	3 + 6	88%	48%	2	55%	55%
BC + 7	3	13%	13%	3 + 7	97%	45%	2	55%	55%
BC + 8	3	13%	13%	3 + 8	82%	33%	2	55%	55%
BC + 8-1	2	13%	57%	3 + 8	82%	86%	2	55%	59%
BC + 9-1	2	57%	57%	3 + 9	86%	55%	2	59%	59%

TABLE 9
Unutilized Resource in the Pool at Each State

State	CPU Left		RAM Left		Storage Left		Network Left	
BC	84	100%	448	100%	2,300	100%	10	100%
BC + 1	76	90%	406	91%	2,034	88%	9	90%
BC + 2	68	81%	364	81%	1,768	77%	8	80%
BC + 3	60	71%	322	72%	1,502	65%	7	70%
BC + 4	52	62%	280	63%	1,236	54%	6	60%
BC + 5	44	52%	238	53%	970	42%	5	50%
BC + 6	36	43%	196	44%	704	31%	4	40%
BC + 7	28	33%	154	34%	438	19%	3	30%
BC + 8	20	24%	112	25%	172	7%	2	20%
BC + 8-1	24	29%	133	30%	338	15%	2	20%
BC + 9-1	16	19%	91	20%	72	3%	1	10%

2 state. Since the allocation policy does not allow de-provisioning unit resources from existing tiers, we are not able to add any more resource to the application tier. In the next Table 9, we show the unutilized resources available at each state starting from initial state to the final state of *BC + 9-1*.

4.2.1 BC State

BC is the state when the business service is in the idle state, with no batch job processing overhead. At the beginning of the *BC* state, the web tier is assigned three unit resources, the application tier is assigned three unit resources and the database tier is assigned two unit resources. Since this is the default configuration when the application is starting, it continues to run in this configuration until one of the tiers reaches the threshold limit. From Table 9, we see that the current pool capacity is at 100 percent for all tiers. From Table 8, we see that the load on the application tier reaches 89 percent. Since the application tier has now exceeded the threshold limit, the CLM orchestrates the resource allocation process. This transitions the system into the *BC + 1* state.

4.2.2 BC + 1 State

At the end of the *BC* state, we noticed that the load on the application tier has exceeded the threshold limit for the tier that resulted in allocating extra resource for the tier. Unlike the traditional resource allocation approach where the cloud management system would uniformly assign unit resources to all the tiers, here only the application tier is allocated an additional unit resource from the resource pool, because it is the only tier whose workload crossed the threshold. The fact to be noticed is that, at end of the *BC + 1* state of the traditional resource allocation approach, the strength of the resource pool was: $\{CPU, RAM, Storage, Network\} = \{48, 257, 1{,}368, 7\}$ which is much less than that of the *BC + 1* state in the case of T-BICA. The *BC + 1* state continues until the load on the application tier crosses the threshold limit again which is to 86 percent. The cloud management layer again starts allocating resource for the application tier and lead the state of the system to *BC + 2*.

We will skip over to the state *BC + 8-1* where the system runs out of resources in the resource pool to allocate unit resource for the application tier. This is where the unique approach of T-BICA lies. In the traditional approach to resource allocation, the system would abandon the allocation process as soon as the pool is exhausted, but T-BICA considers deprovisioning other tiers whose workloads are below their thresholds, and triggers de-provisioning of unit resources from such tiers, while ensuring that the availability of resources at those tiers does not go below their minimum requirements. If it is feasible, then TBICA completes the allocation process and moves on to the next system state. We will be discussing the *BC + 8-1* state and *BC + 9-1* state in the following sections.

4.2.3 BC + 8-1 State

At the end of *BC + 8* state, the load on the application tier reaches 82 percent. This triggers the allocation process of assigning additional resource to the application tier. But the capacity of the resource pool does not permit further allocation as shown in Table 9. To assign one additional unit of resources for the application tier, we would need an additional 94 GB of storage space which we do not have. Now, the T-BICA checks the other tiers for their current workloads and allocation states. Since the minimum requirement of any tier is for two unit resources, we cannot de-provision a unit resource from the database tier. So we check the web tier where the load is 13 percent. According to the Algorithm 3, T-BICA de-provisions one or more unit resources from the web tier and adds the same to the resource pool. So, the resource pool gets an added ζ_w amount of resources from the web tier. Now the CLM is able to assign additional units of resource to the application tier. From Table 8, we see that the load on the application tier has gone up from 82 to 86 percent while the system de-provisions one unit resource from the web tier. Since the web tier has lost one unit resource, the overall load at that tier goes up to 57 from 13 percent. But the provisioning of one unit resource to the application tier brings down the load at that tier and helps keep the system in order in the next state, *BC + 9-1*.

4.2.4 BC + 9-1 State

We reach *BC + 9-1* state from the *BC + 8-1* state by allocating additional resources to the application tier after deprovisioning a unit resource from the web tier. This is the final state for the business service as we are unable to allocate further resources from the free pool. Also, the database and web tiers now have minimum numbers of unit resources needed to meet their objectives. Due to

TABLE 10
Time in Days and Cost in USD at Each State

State	Days	Cost
BC	26.62	$21,542.79
BC + 1	24.75	$20,854.48
BC + 2	22.03	$19,306.18
BC + 3	17.18	$15,627.85
BC + 4	13.92	$13,119.47
BC + 5	12.38	$12,086.53
BC + 6	10.77	$10,872.17
BC + 7	9.81	$10,218.43
BC + 8	8.14	$8,750.85
BC + 8-1	8.46	$8979.08
BC + 9-1	6.87	$7498.19

this, T-BICA is not able to provision any more resources to the business service even though the load on the application tier has again exceeded the threshold limit. The system continues with its present resource allocation until the batch jobs are completed.

We have projected the time in days and the cost associated in USD in Table 10 for each state starting from state *BC* till state *BC + 9-1*. The data in the table helps visualize the reduction in time as we keep provisioning resources for the business service to help to meet the SLA.

In Table 10, we observe a steady decline in the projected time for generating all statements. In the beginning, the initial projected job duration was 26.62 days, which reduces significantly at subsequent state and is finally reduced to 6.87 days in the *BC + 9-1* state. For the static resource allocation approach, the time taken was 19.66 days. Since we presume a seven-day limit for generating all statements, the traditional approach leads to a breach in the SLA between the CSC and CSP. However, T-BICA successfully reduces the time to 6.87 days which is acceptable. Other than job completion time, we have been able to achieve greater savings on costs associated with the resources. At the beginning, the estimated cost was $21,542.79 with the *BC* configuration. At the end of *BC + 9-1* state, the cost is reduced to $7,498.19. For the traditional approach, the final cost was $21,329.44 which is much higher (even without considering possible implications of the SLA breach) than the cost incurred by the T-BICA approach of resource allocation.

In Fig. 6, we see the reduction of job completion times (days), and cost (OpEx) savings at each state. At *BC*, the time is 26.62 days and the cost is $21,542.79. After each state, the projected time to complete the job comes down along with the incurred cost. Ultimately, at the end of the state *BC + 9-1*, the final projected time comes down to 6.87 days which is acceptable and the cost comes down to $7,498.19. This shows major improvement from the traditional approach of resource allocation. In Fig. 7, we show the cost savings and the reduction in time to complete the task.

In Fig. 7, we show the savings of time and cost from initial state to final state. We notice that the savings are highest at *BC + 3* state. Also, there is a drop in savings at the *BC + 8-1* state as we de-provisioned one unit resource from the web tier. But at the end of *BC + 9-1*, we were able to gain 1.6

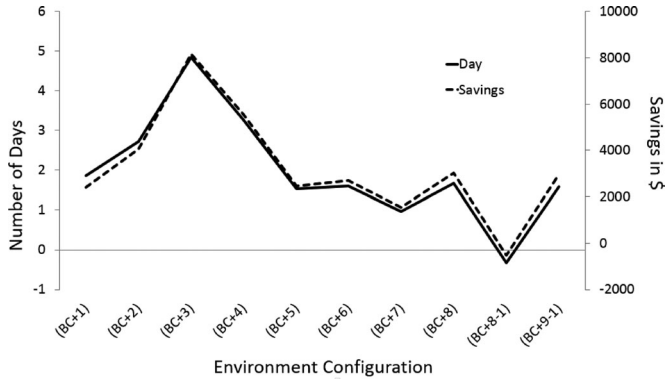

Fig. 6. Reduction in days and cost of the task (T-BICA).

days and saved approximately $1,500 over the *BC + 8-1* state. From *BC + 9-1* state and onwards, there are no changes in the savings as we are not allocating any more resources to the business service.

5 Conclusion

Using the example of an enterprise grade bank account management system, we have been able to demonstrate the differences between the traditional static resource allocation approach in a modern private cloud environment, and our simple but unique analytical T-BICA approach of resource allocations at individual tiers of the business service—not only considering the overall business service SLOs but also the actual IT resource utilization at each tier. We have shown that the traditional resource allocation approach is costly and takes longer to finish critical jobs, since the CLM allocates the required additional resources in a static manner across all tiers without monitoring individual tier resource utilization.

By contrast, T-BICA has the ability to monitor individual tier resource utilization and allocate resources to only those tiers where the load has exceeded a threshold. Also, in the static allocation approach, the allocation process halts when the resource pool is completely exhausted (which may happen after just a few iterations, because of wasted allocations to under-loaded tiers). But in T-BICA, it is possible utilize capacity in the resource pool better and de-provision resources from under-utilized tiers to assign them to over-loaded tiers of the business service. T-BICA ensures the resource de-provisioning process does not

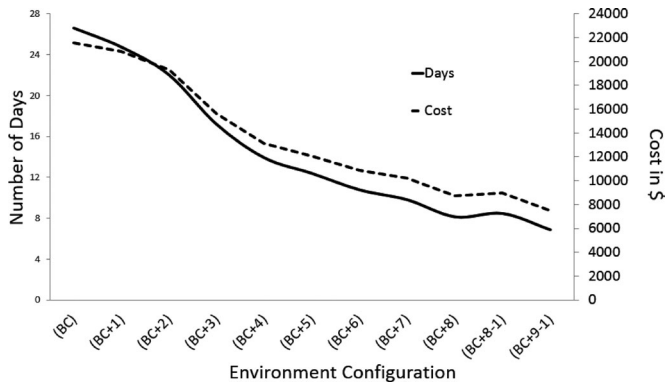

Fig. 7. Savings in days and cost of the task (T-BICA).

breach the minimum resources limits of any tier. In our realistic experiment, T-BICA saved 65.19 percent in cloud subscription (OpEx) costs and completed all the batch jobs in 6.87 days, reducing the time to deliver by 74.18 percent.

T-BICA makes for justified, cost-effective, and efficient resource allocation for all tiers of a multi-tier cloud-based business service. A cloud provider that adopts the T-BICA approach will find it easier to meet its SLOs, and avoid breaching its SLAs, while reducing the OpEx of cloud subscription for CSCs. T-BICA also helps cloud subscribers reduce undetected server sprawl for enterprise business services hosted in cloud environments by avoiding wasted resource provisioning by a static orchestration process. It thus also helps promote sustainability (by lowering resource costs) and is a "green" approach in a significant way.

Acknowledgments

A brief extended abstract [1] of this work, without detailed analyses or results, was presented at the 6th Annual IEEE International Systems Conference (IEEE SysCon 2012), Vancouver, Canada, in March 2012. Shrisha Rao is the corresponding author.

References

[1] M. F. Mithani and S. Rao,"Improving resource allocation in multi-tier cloud systems," in *Proc. 6th Annu. IEEE Int. Syst. Conf.*, Vancouver, BC, Canada, Mar. 2012, pp. 1–6.

[2] P. Mell and T. Grance. (2011, Sep.). The NIST definition of cloud computing, NIST Special Publication 800-145, Tech. Rep. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[3] Y. Song, Y. Sun, and W. Shi, "A two-tiered on-demand resource allocation mechanism for VM-based data centers," *IEEE Trans. Services Comput.*, vol. 6, no. 1, pp. 116–129, 1st Quarter 2013.

[4] M. Kesavan, I. Ahmad, O. Krieger, R. Soundararajan, A. Gavrilovska, and K. Schwan, "Practical compute capacity management for virtualized datacenters," *IEEE Trans. Cloud Comput.*, vol. 1, no. 1, pp. 88–100, Jan.–Jun. 2013.

[5] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1107–1117, Jun. 2013.

[6] C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. Cervello-Pastor, and A. Monje, "On the optimal allocation of virtual resources in cloud computing networks," *IEEE Trans. Comput.*,vol. 62, no. 6, pp. 1060–1071, Jun. 2013.

[7] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE Trans. Services Comput.*, vol. 5, no. 2, pp. 164–177, Apr.–Jun. 2012.

[8] F. Teng and F. Magouls, "A new game theoretical resource allocation algorithm for cloud computing," in *Proc. 5th Int. Conf. Adv. Grid Pervasive Comput.*, 2010, vol. 6104, pp. 321–330.

[9] G. Wei, A. Vasilakos, Y. Zheng, and N. Xiong, "A game-theoretic method of fair resource allocation for cloud computing services," *J. Supercomput.*, vol. 54, no. 2, pp. 252–269, 2010.

[10] P. S. Pillai and S. Rao, "Resource allocation in cloud computing using the uncertainty principle of game theory," IEEE Syst. J. 2015, DOI: 10.1109/JSYST.2014.2314861.

[11] L. Wu, S. Garg, and R. Buyya, "SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments," in *Proc. 11th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, May 2011, pp. 195–204.

[12] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Proc. Grid Comput. Environ. Workshop*, 2008, pp. 1–10.

[13] M. Brock and A. Goscinski, "Grids vs. clouds," in *Proc. 5th Int. Conf. Future Inf. Technol.*, 2010, pp. 1–6.

[14] P. Samimi and A. Patel, "Review of pricing models for grid amp; cloud computing," in *Proc. IEEE Symp. Comput. Inf.*, 2011, pp. 634–639.

[15] T. W. Shinder, "Private cloud reference model, Microsoft Inc,'' 2011.

[16] Data center site infrastructure tier standards: Topology, Uptime Institute Professional Services, 2009.

[17] A. Goscinski and M. Brock, "Toward higher level abstractions for cloud computing," *Int. J. of Cloud Comput.*, vol. 1, no. 1, pp. 37–57, 2011, DOI:10.1504/IJCC.2011.043245.

[18] L. Badger, T. Grance, R. Patt-Corner, and J. Voas. (2012, May). Cloud computing synopsis and recommendations, NIST Special Publication 800-146 [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-146/sp800-146.pdf

[19] M. F. Mithani, M. A. Salsburg, and S. Rao,"A decision support system for moving workloads to public clouds," in *Proc. Annu. Int. Conf. Cloud Comput. Virtualization*, Singapore, May 2010, pp. 3–9.

[20] S. Bose, M. Salsburg, and M. Mithani, "Leveraging smart-meters for initiating application migration across clouds for performance and power-expenditure trade-offs," in *Proc. 1st IEEE Int. Conf. Smart Grid Commun.*, 2010, pp. 102–107.

[21] Q. Zhu and G. Agrawal, "Resource provisioning with budget constraints for adaptive applications in cloud environments," in *Proc. 19th ACM Int. High Performance Distrib. Comput.*, 2010, pp. 304–307.

[22] P. Lama and X. Zhou, "Efficient server provisioning with control for end-to-end response time guarantee on multitier clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 1, pp. 78–86, Jan. 2012.

[23] J. Jiang, J. Lu, G. Zhang, and G. Long, "Optimal cloud resource auto-scaling for web applications," in *Proc. 13th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, Delft, The Netherlands, May 2013, pp. 58–65.

[24] S. Wagner. (2013, Jun.). The great hope of cloud economics and the over-provisioning epidemic, *Cloudyn White Paper* [Online]. Available: http://cloudyn.com/wp-content/uploads/2013/06/The_Great_Hope_of_Cloud_Economics.pdf

[25] F. Chang, J. Ren, and R. Viswanathan, "Optimal resource allocation in clouds," in *Proc. 3rd IEEE Int. Conf. Cloud Comput.*, 2010, pp. 418–425.

[26] B. An, V. Lesser, D. Irwin, and M. Zink, "Automated negotiation with decommitment for dynamic resource allocation in cloud computing," in *Proc. 9th Int. Conf. Auton. Agents Multiagent Syst.: Volume 1*, 2010, pp. 981–988.

[27] A. Rai, R. Bhagwan, and S. Guha, "Generalized resource allocation for the cloud," in *Proc. 3rd ACM Symp. Cloud Comput.*, 2012, pp. 15:1–15:12.

[28] T. Saaty,, "What is the analytic hierarchy process?" in *Mathematical Models Decision Support*, series NATO ASI Series, G. Mitra, H. Greenberg, F. Lootsma, M. Rijkaert, and H. Zimmermann, Eds., vol. 48. Berlin, Germany: Springer, 1988, pp.109–121.

[29] D. Ergu, G. Kou, Y. Peng, Y. Shi, and Y. Shi, "The analytic hierarchy process: Task scheduling and resource allocation in cloud computing environment," *J. Supercomput.*, vol. 64, no. 3, pp. 835–848, 2013.

[30] R. Han, M. M. Ghanem, L. Guo, Y. Guo, and M. Osmond. Enabling cost-aware and adaptive elasticity of multi-tier cloud applications. *Future Gener. Comput. Syst. 32*, vol. 32, pp. 82–98, Mar. 2014.

[31] H. Goudarzi and M. Pedram, "Multi-dimensional SLA-based resource allocation for multi-tier cloud computing systems," in *Proc. IEEE 4th Int. Conf. Cloud Comput.*, 2011, pp. 324–331.

[32] UK Cloud Adoption Trends: Current Landscape & Future Outlook, Cloud Security Alliance. (2012) [Online]. Available: http://tinyurl.com/Ccsk-UK-Cl-Adptn

[33] J. McKendrick. (2011, Nov.). *Enterprise Advances into the Cloud: 2011 IOUG Cloud Computing Survey*. Unisphere Res., [Online]. Available: HTTP://TINYURL.COM/78HCK4X

[34] *The Cloud Dividend: Part One.* (2010, Dec.). Centre for Economics and Business Research Ltd. [Online]. Available: http://tinyurl.com/7grfxzh

[35] (2014) Amazon Web Services, Cloud Pricing Model [Online]. Available: http://aws.typepad.com/aws/price-reduction/

[36] A. Mehta, M. Menaria, S. Dangi, and S. Rao, "Energy conservation in cloud infrastructures," in *Proc. 5th Annu. IEEE Int. Syst. Conf.*, Montreal, QC, Canada, Apr. 2011, pp. 456–460.

[37] (2014) Rackspace, Cloud Pricing Model [Online]. Available: http://www.rackspace.com/cloud/servers/pricing/
[38] D. Warneke and O. Kao, "Exploiting dynamic resource allocation for efficient parallel data processing in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 6, pp. 985–997, Jun. 2011.
[39] Y. Yazir, C. Matthews, R. Farahbod, S. Neville, A. Guitouni, S. Ganti, and Y. Coady, "Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis," in *Proc. 3rd IEEE Int. Conf. Cloud Comput.*, 2010, pp. 91–98.
[40] J.-P. Brans and B. Mareschal, "Promethee methods," in *Multiple Criteria Decision Analysis: State of the Art Surveys*, series International Series in Operations Research & Management Science, vol. 78. New York, NY, USA: Springer, 2005, pp. 163–186.
[41] D. Menasce and V. Almeida, *Capacity Planning for Web Services: Metrics, Models, and Methods*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2002.
[42] X. Liu, J. Heo, L. Sha, and X. Zhu, "Adaptive control of multi-tiered web applications using queueing predictor," in *Proc. 10th IEEE Netw. Oper. Manag. Symp.*, 2006, pp. 106–114.
[43] X. Wang, Z. Du, Y. Chen, and S. Li, "Virtualization-based autonomic resource management for multi-tier web applications in shared data center," *J. Syst. Softw.*, vol. 81, no. 9, pp. 1591–1608, 2008.
[44] P. Padala, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem, "Adaptive control of virtualized resources in utility computing environments," in *Proc. 2nd ACM SIGOPS/EuroSys Eur. Conf. Comput. Syst.*, 2007, pp. 289–302.
[45] W. Stromquist, "How to cut a cake fairly," *Amer. Math. Monthly*, pp. 640–644, Oct. 1980.
[46] S. J. Brams and A. D. Taylor, *Fair Division: From Cake-Cutting to Dispute Resolution*. Cambridge, U.K.: Cambridge Univ. Press, 1996.
[47] D. Fudenberg and J. Tirole, *Game Theory*. Cambridge, MA, USA: MIT Press, 1991.
[48] (2014) Amazon Web Services, Pricing Plans [Online]. Available: https://aws.amazon.com/premiumsupport/pricing/

Jyotiska Nath Khasnabish received the MTech degree in information technology from IIIT-Bangalore, a graduate school of information technology in Bangalore, India. He is a data engineer specializing in products and strategy at DataWeave. His research interests are in distributed computing applied to big data and massive online systems.

Mohammad Firoj Mithani is an Enterprise Cloud Service lead designer based in Melbourne, Australia, who has more than 15 years of hands-on work experience in leading and designing complex business critical IT systems and support processes for the Fortune 100 BSFI, healthcare and telecom organizations around the world. He is a PMI certified professional (PMP) and an EMC certified Cloud Architect (EMCCA). His core strengths are in designing business-centric smart cloud platforms to optimize return on investment on IT system deployment, utilization and operations. He is an inventor of three patent-pending workload placement analytic tools in the field of cloud computing. He is a member of the IEEE.

Shrisha Rao received the MS degree in logic and computation from Carnegie Mellon University, and the PhD degree in computer science from the University of Iowa. He is an associate professor at IIIT-Bangalore. His research interests are in distributed computing, specifically algorithms and approaches for concurrent and distributed systems, and include solar energy and microgrids, cloud computing, energy-aware computing ("green IT"), and demand side resource management. He is a senior member of the IEEE and a member of the IEEE Computer Society, the ACM, the American Mathematical Society, and the Computer Society of India.