Postprint of: Orzechowski P., Krawczyk H. (2023). General Provisioning Strategy for Local Specialized Cloud Computing Environments. In: Zamojski W., Mazurkiewicz J., Sugier J., Walkowiak T., Kacprzyk J. (eds) Dependable Computer Systems and Networks. DepCoS-RELCOMEX 2023. Lecture Notes in Networks and Systems, vol 737. Springer, Cham. DOI: 10.1007/978-3-031-37720-4_18

General provisioning strategy for local specialized cloud computing environments

Piotr Orzechowski[1][0000-0002-4339-9707] and Henryk Krawczyk[2][0000-0003-0436-6264]

[1] Centre of Informatics Tricity Academic Computer and Network, Gdansk University of Technology, Gabriela Narutowicza 11/12, 80-233 Gdansk, Poland,
porzechowski@task.gda.pl

[2] Faculty of Electronics, Telecommunications and Informatics, Gdansk University of Technology Gabriela Narutowicza 11/12, 80-233 Gdansk, Poland,
hkrawk@pg.edu.pl

Abstract. The well-known management strategies in cloud computing based on SLA requirements are considered. A deterministic parallel provisioning algorithm has been prepared and used to show its behavior for three different requirements: load balancing, consolidation, and fault tolerance. The impact of these strategies on the total execution time of different sets of services is analyzed for randomly chosen sets of data. This makes it possible to improve the project and to implement the proper strategies for the local TASKcloud environment used in our institution.

Keywords: cloud computing, management strategy, criteria of optimalizations, provisioning algorithms.

1 Introduction

In general, service providers establish a service level agreement (SLA [1]) covering the general terms and conditions in which they will work with customers. The SLA is not only a set of conditions for service providers, but it could be also a source of benefits for customers. The contract between the provider and customer describes different characteristics of the service, which makes the services comparable between different providers. The SLA should also contain methods of redressing service issues. Other topics mentioned in SLA documents include:

- client expectations according to his/her needs,
- detailed descriptions of every service offered, under all possible circumstances, with the turnaround times included,
- definition of quality measurement metrics and quality level assurance,
- compensation or payment if the provider cannot properly fulfill this SLA.

Cloud computing [2] is mainly built on top of virtualization, as cloud users typically rent virtual resources from cloud providers. A popular form of virtualization is the use

of virtual units (virtual machines, containers) which are created to run on a host machine (typically a physical server). Thanks to this, cloud architectures integrate IT environments and share scalable resources across a network to deliver an online platform on which client applications can run. In general, cloud systems are highly complex, as they deal with a range of distributed components, users, and deployment scenarios.

In the literature, different enhancement and approaches to cloud management are considered and various models are proposed [4]. In general, the service provider is responsible for managing the resources to fulfill the requests generated by users. Service providers employ suitable algorithms to manage the incoming client requests (services) and to manage their virtual resources efficiently. Management strategies make it possible for providers to maximize revenue by utilizing their available resources up to their limits. In practice, in terms of the performance of cloud computing resources, the choice of management strategy makes a pronounced difference.

Our consideration and experiments focus on the implementation of a provisioning algorithm for local cloud computing, with the assumption that these local providers possess a more limited number of services and resources available for clients. Typical local cloud architectures are implemented through commonly available open-source software (Unix, OpenStack, Kubernetes). The presented models have been simulated hosting the TASKcloud [5] service (based on OpenStack software) which is a cloud computing service developed and deployed in our institution. The paper focuses on some aspects related to the main management strategies regarding the allocation and provisioning of resources (virtual units). They can be defined in different ways under the accepted assumptions related to clients' requirements, cloud architecture, models of services and resources, and optimization criteria. Resource Allocation refers to the allocation (reservation) of a pool of resources represented by virtual machines or containers (virtual resources – VR) to satisfy the SLA previously accepted by both the user and the cloud provider, while Resource Provisioning is the effective provisioning of a portion of the reserved resources to execute the fixed set of services notified by the user. Fig. 1 explains the proposed approach. When a cloud provider accepts a request from a customer, it has to create the appropriate number of VRs and allocate user services to run. A typical example of resource provisioning is the deployment of a new virtual machine by the consumer, which uses a subset of the physical resources to run the single service. Due to virtualization, we can allocate resources flexibly, based on the current client demands. Then we can estimate the optimal values of resources for a concrete client demand. In general, this will be a much lower value than the value estimated based on the SLA.

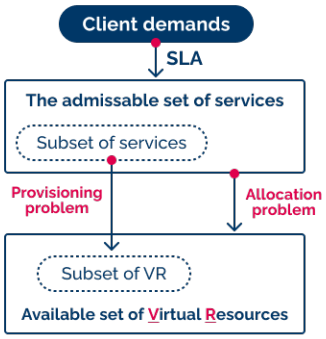

Fig. 1. Considered scope of cloud management (source: authors).

The paper considers local specialized clouds, such as the mentioned TASKcloud, and considers a resource provisioning strategy for a lower number of services and resources compared to the global clouds deployed by such Big Tech firms such as Google, Microsoft and IBM. In such a case, deterministic provisioning algorithms can be considered. Their behavior can be examined for different optimization criteria, i.e., load balancing, consolidation, and fault tolerance. The impact of many parameters related to heuristic algorithms can be eliminated this way. Moreover, a parallel version of the algorithm to increase its speed and to consider mixed optimization criteria has been prepared. The paper presents that mixed criteria could have a significant impact on the execution time compared to a single criterion. It makes it possible to estimate the assurance cost of t-faults tolerance, where t is the maximal number of faulty virtual units in the cloud.

The structure of the article is as follows. First, in Section 2, the state-of-the-art in the field of the practical implementation of management strategies in local cloud environments is discussed. In Section 3, the model of the assumed provisioning problem is described. Three criteria and the universal parallel algorithm to solve load balancing, consolidation, and fault tolerance problems are defined. The experiments are described, and the test results of the algorithm are discussed in detail in Section 4. Section 5 concludes the article.

2 Categories of cloud provisioning

Organizations can manually provide whatever resources and services they need, but public cloud providers offer tools to provision multiple resources and services, for instance: AWS CloudFormation, Microsoft Azure Resource Manager, and Google Cloud Deployment Manager. Such solutions concern global clouds and primarily concentrate on the administrative problems of cloud provisioning and offer many tools to support whole organizations rather than single customers. This paper focuses on local specialized clouds and considers the methods of allocating a cloud provider's resources and services to customers. In the current considered provisioning problems, the customer signs a formal contract of service with the cloud provider [7]. The provider prepares

the agreed-upon resources or services for the customer and delivers them. It is a process that can be conducted using one of three delivery models. Each delivery model differs depending on the kinds of resources or services the organization purchases, how and when the cloud provider delivers those resources or services, and how the customer pays for them. The three models are advanced provisioning, dynamic provisioning, and user self-provisioning:

- Advanced Provisioning – the customer requests services from the provider and the provider prepares the appropriate resources in advance. The customer is charged a flat fee or is billed monthly,
- Dynamic provisioning – the customer can purchase cloud resources based on average consumption needs. The cloud provider deploys and adjusts the resources to match the customer's usage demands. Based on the customer's fluctuating demands, the provider allocates more resources when they are needed and removes them when they are not. Cloud deployments typically scale up to accommodate spikes in usage and scale down when demand decreases. The customer is billed on a pay-per-use basis. When dynamic provisioning is used to create a hybrid cloud environment, it is sometimes referred to as cloud bursting,
- User self-provisioning (also called cloud self-service) – the customer buys resources from the cloud provider through a web interface or portal and the cloud provider makes these resources available shortly after purchase. This usually involves creating a user account and paying for the resources with a credit card. Those resources are then quickly spun up and made available for use – within hours, if not minutes. Examples of this type of cloud provisioning include an employee purchasing cloud-based productivity applications via, e.g., the Microsoft 365 suite or G Suite.

A self-service provisioning model helps to streamline users' requests and manage cloud resources but requires strict rules to ensure they do not provision resources they should not. In this paper, the focus is on this type of provisioning model. The following metrics can be distinguished within the above provisioning approach:

- Scalability – there is no requirement for forecasting infrastructure needs; organizations can simply scale up and scale down their cloud resources based on short-term usage requirements,
- Provisioning speed – developers can quickly spin up a set of workloads on demand, removing the need for an IT administrator who provisions and manages the compute resources,
- Cost savings – many cloud providers allow customers to pay for only what they consume. However, the attractive economics presented by cloud services can present its own challenges, which organizations should address in their cloud management strategies.

It is a well-known fact that resource over-provisioning can cost users more than necessary and resource under-provisioning hurts application performance. In general, it is a complicated optimization problem, and there is a wide research avenue available for solving this. In the paper [8], some details about various optimization techniques for resource provisioning are presented. It has been shown that the cost-effectiveness of

cloud computing strongly depends on how well the customer can optimize the cost of renting resources (Virtual Machines) from cloud providers. In the above paper, a framework is proposed. It is inspired by a cloud layer model, to enable the optimal provision of resources by combining the concepts of autonomic computing, linear regression and Bayesian learning. The efficacy of the proposed framework is evaluated using both the CloudSim toolkit and real-world workload traces from Google, followed by the traces from Clarknet. Such parameters as response time, SLA violations, virtual machine usage hours and cost were evaluated. In the paper [9], the authors propose a concrete solution for migrating physical servers to a cloud with the usage of the Azure cloud framework. The utilization of physical server resources on remote VM servers is considered. Such a migration process in the framework was implemented in two phases: first by integrating physical servers into virtual ones by creating virtual machines, and then by integrating the virtual servers into cloud service providers in a cost-effective manner. Two virtual machine instances were created using Microsoft Hyper-V on Windows Server 2016 R2. Applications that were installed on a workstation were migrated to the VM and the performance of this VM was monitored using a PowerShell script. Then Tableau was used to generate load and do analytical calculations to evaluate the physical server functionality.
The above papers concentrate on the IaaS level, considering VM-based environments, where a hypervisor will strictly allocate resources to the deployed VMs. The deployed VMs, however, can compete for the shared physical resources, but the hypervisor should detect and prevent this to not violate SLA requirements. In this paper, a more general approach is proposed to mitigate these constraints, where cloud services are assigned to virtual units. A dynamic provisioning approach is presented, and a deterministic algorithm is proposed. As was mentioned in Section 1, three different optimalization criteria are analyzed and some results are given. Moreover, the implementation of the algorithm is prepared in such a way that it can be used in the TASKcloud environment. Deployment of TASKcloud is fully automated using advanced Ansible playbooks and it can be adapted to change the scheduling mechanisms.

3 Model of cloud environments to optimize provisioning strategies

As was shown in the previous section, there are many proposals on how to build a suitable provisioning strategy for cloud environments. However, heterogenous resources and the different methods of their cooperation, as well as the diversity, variability, and unpredictability of the required workload, and different needs of various cloud users make universal, simple, and effective methods most useful. They can be formulated based on general models, which can be used at different levels of cloud management strategies.

Consider a set P of permutations of services and virtual resources; their number can be $P = n * m$, where m is the number of all possible services and n is the number of all possible virtual resources. For the exact solution of the provisioning problem, all possible allocation modes must be reevaluated and the best mode chosen. Due to the large number of exponential modes, the problem is an example of a set packing problem

which is of NP-complete type. Moreover, in the proposed method, the provisioning of a set of services at one point in time is considered instead of queued services that are provisioned one by one. The assumed notation is presented below:

- $S = \{s_1, s_2, \dots, s_m\}$ – is the set of user demands representing services waiting to run in the computing cloud, where $s_i, i = 1,2, \dots, m$ is a user demand to run the $i - th$ service. It can be a single task or a scenario of tasks.
- $R = \{r_1, r_2, \dots, r_n\}$ – is the set of virtual resources available in the cloud, where r_j represents the $j - th$ resource which belongs to one of the cloud servers. Each virtual resource is supported by some physical resources described by computing capabilities, such as computational power, storage, and cloud services.
- $\psi(S, R)$ – is the allocation matrix of required services S to the available cloud resources belonging to R, in brief:
  $-\ \psi = [\psi_{ij}]$, where:

$$\psi_{ij} = \begin{cases} 1 - if\ service\ s_i\ is\ allocated\ to\ cloud\ resource\ r_j \\ 0 - otherwise \end{cases} \quad (1)$$

  where $i \epsilon \{1,2, \dots, m\} \wedge j \epsilon \{1,2, \dots, n\}$

- δ – is the vector representing the current rate of use of all cloud resources R, before allocation of the new services from S, i.e., $\delta = [\delta_1, \delta_2, \dots, \delta_n]$, $\delta_i \in < 0,1.2 >$. In further considerations, a small 20% over-provisioning is allowed.

In practice, the rate of use, as a ratio of the amount of currently occupied resources to the amount of all resources available, can be calculated. In simple cases, the percentage of virtual machines currently active to all available space can be used here.

Let $\gamma = [\gamma_{ij}]$ be the vector determining the rate of the $j - th$ resource use when the $i - th$ service can be assigned to it; $\gamma_{ij} \in < 0,1 >$. Note that γ can be calculated in the same way as δ. In practice, it means the percentage of engaged resources.

The load of the *j-th* resource after the allocation of services S on resources R according to ψ can be calculated in the following way:

$$\delta'_j = \delta_j + \sum_{i=1}^{m} \psi_{ij} \cdot \gamma_{ij} \quad (2)$$

Let T be the matrix of the given execution times for all services running on all resources, i.e.:

$$T = [t_{ij}] \quad (3)$$

where t_{ij} denotes the processing time of service s_i on resource r_j. It can be calculated empirically either by testing or by estimating the service properties and the characteristics of the resources.

Note that it is true where $\gamma_{ij} \leq 1 - \delta_j$, otherwise the processing time t_{ij} can be increased according to the following formula:

$$t'_{ij} = \left(\delta_j + \gamma_{ij}\right) \cdot t_{ij} \tag{4}$$

The execution time of all services S running on resources R for allocation $\psi(S,R)$ is denoted by $\tau(S,R)$. If the services are to be processed sequentially, then:

$$\tau(S,R) = \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij} \cdot \psi_{ij} \tag{5}$$

For the parallel execution of all services (which is possible when $n \geq m$):

$$\tau(S,R) = max_{i=1,..,m} \sum_{j=1}^{n} t_{ij} \cdot \psi_{ij} \tag{6}$$

In general:

$$max_{i=1,..,m} \sum_{j=1}^{n} t_{ij} \cdot \psi_{ij} \leq \tau(S,R) \leq \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij} \cdot \psi_{ij} \tag{7}$$

Let us consider provisioning problems for three different optimization criteria: load balancing, consolidation, and fault tolerance. Load balancing algorithms are used to distribute new demands of users (services) among the virtual resources to guarantee a well-balanced load across all cloud nodes. In contrast, consolidation is usually achieved by spreading the service workload over a smaller set of resources so the servers remaining unused can be powered down or put into standby mode. The first approach minimizes the total execution time of the set of services, in other words, to maximize the use of their resources at a lower overall client cost to increase their profit. The second approach copes better with highly fluctuating demands from clients. Moreover, having a set of free servers (nodes) that is not currently needed also makes it possible to design fault-tolerant systems. In that case, we plan the execution of each of the user tasks on more than one node. Below, we discuss the criteria in more detail and provide formal optimization criteria.

Load balancing algorithms are used to distribute new requests of users (services) in a cloud between the virtual units to guarantee an equal number of services allocated to each cloud server. However, each client demand can be expressed by service workloads to run on virtual units. Then, load balancing is a mechanism to balance the load by uniformly distributing the workload among the nodes [10]. Effective load balancing mechanisms will optimize the utilization of resources and improve the cloud's performance. There are various implementations of such mechanisms based on different load balancing algorithms [11]. In [12], capacity planning methods for cloud users and cloud service providers, and algorithms that combine the capabilities of different strategies which are more efficient, are considered. In consequence, load balancing algorithms seek to distribute service workloads across several virtual machines in a manner that minimizes the average time taken to complete the execution of those workloads, which typically results in server utilization being maximized and balanced.

The optimization problem for load balancing is defined as looking for (S,R), which minimizes $\tau(S,R)$, subject to:

$$\begin{cases} \delta_i \cong \delta_j \, \forall_{i \neq j} \, i, j \in \{1, 2 \ldots, \mathrm{n}\} \\ \sum_{i=1}^{m} \psi_{ij} \geq 0 \\ \sum_{j=1}^{n} \psi_{ij} = 1 \\ \gamma_{ij} \leq 1 - \delta_j \end{cases} \tag{8}$$

Workload consolidation aims at maximizing the usage of servers by grouping services to run concurrently on fewer virtual units. Workload consolidation is one way to reduce resource wastage by clustering services on a subset of the pool of available machines. This technique is used to maintain control over the potentially high economic and environmental cost [13]. Many different approaches have been proposed for workload consolidation, but it is unclear which of the proposed approaches works best in each situation. In the paper [14], the authors showed that consolidation algorithms, whose goal is to maximize the number of empty physical machines, perform many virtual machine migrations, named eager migrations. These migration processes have a significant impact on the response times of the services deployed on those machines. The authors propose a new method and a heuristic to decide which virtual machines should be migrated. This solution takes into account the variability of the sizes of the virtual machines and prioritizes virtual machines with a steady capacity to be migrated first. In the paper [15], the authors proposed a solution to allocating a set of services based on a bin packing problem. The described framework is a semi-online workload management system which gathers incoming user requests to start a workload and packages them into sets. Then a whole group of services is taken into account during the allocation process. Such an allocation policy produces a saving of up to 40% of the resources compared to other consolidation algorithms.
For consolidation, the optimalization criterion of maximizing the number of empty nodes (value |I|) is proposed:

$$I = \left\{ j \middle| \psi_{ij} = 0 \, \forall_i \right\}, \text{|I| is cardinality of } I \tag{9}$$

subject to:

$$\gamma_{ij} \leq 1 - \delta_j \tag{10}$$

$$\forall j, j \in \{1,2, \ldots, \mathrm{n}\} \backslash \mathrm{I} \; \sum_{i=1}^{m} \psi_{ij} \geq 1 \tag{11}$$

$$\forall j, j \in I \sum_{i=1}^{m} \psi_{ij} = 0 \tag{12}$$

A fault-tolerant system works on one of two strategies. The replication strategy assumes that service replicas are running for each service in parallel and the result is obtained by majority voting. Alternatively, the redundant strategy assumes that the redundant servers or virtual units reside on an inactive mode unless and until any fault tolerance system demands their availability. Thus, if one part of the system fails, it has other instances that can be used in its place to keep it running. Extensive research efforts are consistently being made to implement fault tolerance in cloud infrastructures: the paper [16] gives a systematic and comprehensive elucidation of different fault types, their causes and various fault tolerance frameworks used in cloud implementations. Recently, cloud computing-based environments have presented new challenges to support fault-tolerance and opened up new paths to develop novel strategies, architectures, and

standards. In the paper [17], the needs and solutions of fault-tolerance in cloud computing are discussed and future research directions specific to the development of cloud computing fault-tolerance are enumerated. In further considerations, an assumption has been made that t fault tolerance means that resources are allocated in a manner such that the impact of t failures (e.g., failures of virtual units) on the system performance is minimized or unimportant. The optimalization problem for t-faults tolerance, where $n \geq 2t + 1$ can be defined as minimizing $\tau(S, R)$, can be expressed as:

$$\sum_{j=1}^{n} \psi_{ij} \geq t + 1, \forall_i \in \{1,2, \dots, m\} \tag{13}$$

$$\delta'_j \leq 1.2 \tag{14}$$

As can be seen, each optimalization problem given above can be solved either separately or in different combinations, depending on the user needs. There are many available options based on genetic algorithms or artificial intelligence. They differ in some assumptions related to the features of the user requests and the services and capabilities of the cloud resources. In general, to minimize the total running time, the following properties are considered:

1. For user needs – requirements contained in SLA agreements should be considered during management processes and their implementation requires consideration at three levels: global, local, and operating system level. In this report, we investigate the local level,
2. For services – they are deterministic, their processing time preemptive without precedence constraints regarding the order of services, and each service cannot be further split into smaller subtasks,
3. For resources – the processing capacity of the node remains unchanged but bounded, i.e., a limited number of services can be processed in sequential order of provisioning. The number of resources (nodes) can be invariant according to the user needs.

4 Experiments and results

To evaluate the provisioning strategy, the following parallel provisioning algorithm is proposed:

Algorithm

```
Input Data: S,R,δ,T

do in parallel:
  create all allocations for ψ(S,R)
  select allocations satisfying criteria (8), (9), (13)
end
make selection of the best allocations
```

```
Output Data: ψ(S,R) and τ(S,R) according to the selected
criteria
```

Because the above algorithm is NP complete type, despite the proposed parallelization, only configurations with a maximum of 8 services and 4 resources (nodes) have been analyzed. It has been assumed that each service has its copy, which is a backup in case of a node failure. Services are named s_i and their copies are named s_i'. Let us assume the following input data for 8 services:

Table 1. Values of matrix T

Services/Resources	r_1	r_2	r_3	r_4
s_1	4	5	3	4
s_1'	4	5	3	4
s_2	3	4	3	2
s_2'	3	4	3	2
s_3	5	4	6	3
s_3'	5	4	6	3
s_4	3	3	4	3
s_4'	3	3	4	3

Table 2. Values of matrix γ

Services/Resources	r_1	r_2	r_3	r_4
s_1	0.4	0.5	0.3	0.4
s_1'	0.4	0.5	0.3	0.4
s_2	0.3	0.4	0.3	0.2
s_2'	0.3	0.4	0.3	0.2
s_3	0.5	0.4	0.6	0.3
s_3'	0.5	0.4	0.6	0.3
s_4	0.3	0.3	0.4	0.3
s_4'	0.3	0.3	0.4	0.3

The solutions presented in Table 3 were obtained using the proposed algorithm.

Table 3. Optimal allocation for the considered model and 8 services. Values of processing time are in brackets. (x^*) means acceptation of resource overload.

Criteria/Resources	r_1	r_2	r_3	r_4
Load balancing (LB)	$s_4, s_4'(6)$	$s_3'(4)$	$s_1, s_1'(6)$	$s_2, s_2', s_3(7)$
Consolidation (CONS)			s_1, s_1', s_2, s_2' (12.6*)	s_3, s_3', s_4, s_4' $(12.6*)$
Fault tolerance (FT)	$s_1', s_4'(7)$	$s_3'(4)$	$s_1, s_2'(6)$	$s_2, s_3, s_4(8)$
Load balancing & Fault tolerance (LBFT)	$s_1', s_4'(7)$	$s_3'(4)$	$s_1, s_2'(6)$	$s_2, s_3, s_4(8)$
Consolidation & Fault tolerance (LBFT)	$s_1, s_3, s_4(12.6*)$		$s_1', s_2, s_4'(10)$	$s_2', s_3'(5)$

Experiments were run for sets of 2, 4, 6, 8, 10 and 12 services and the results are presented in Fig. 2 and Fig. 3.

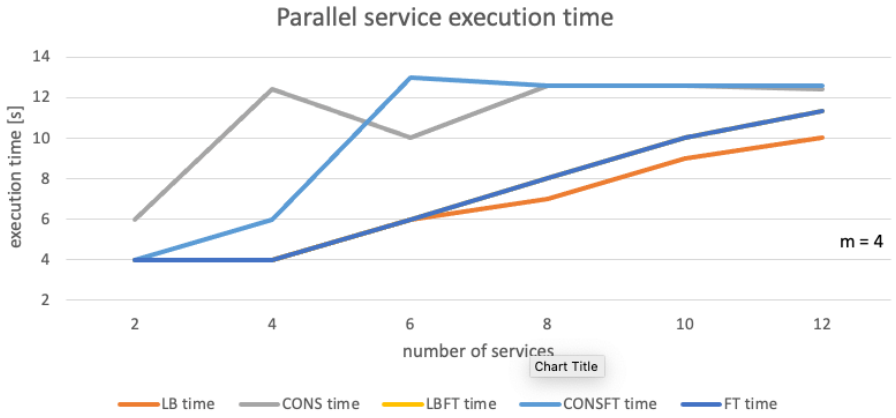

Fig. 2. Execution time of a set of services on different resources provisioned using different algorithms.

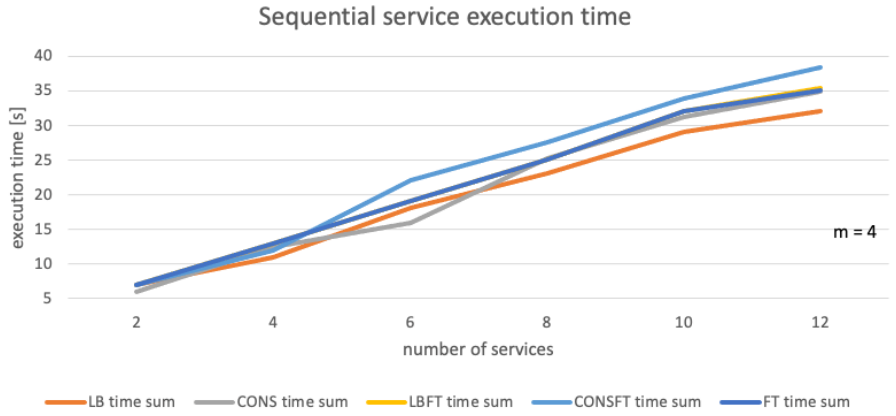

Fig. 3. Sum of execution times of services on different resources provisioned using different algorithms.

The randomly chosen values of Table 1 and Table 2 were analyzed and the processing times of the sets of services were determined. It is shown that the most time-consuming criterion is fault tolerance, the second is consolidation, while the lowest processing time is consumed for the load balancing criterion. Moreover, the common consideration of the two criteria of load balancing and fault tolerance produced a slightly better result

than the common consideration of consolidation and fault tolerance, which is also verified in practice. This provides a practical suggestion for the implementation of permission strategies.

5 Final Remarks

Three different cases of provisioning problems have been considered. The execution time of a set of independent services has been compared. A formal model that can be used at different levels of cloud resources – virtual units or physical units – has been proposed. Three optimalization problems have been classified based on their mean processing time. A hybrid approach has been investigated, and load balancing with fault tolerance is shown to produce more promising results than consolidation with fault tolerance. The presented model makes it possible to analyze a series of service sets required to run in a cloud environment and achieve acceptable scalability. It makes it possible to determine the proper strategy of provisioning for changing user requirements or clients' demands in near real time. The sequential and parallel execution of services on one node can also be considered. To improve the provisioning speed of much bigger sets of services and resources (which will be interesting for global clouds), a heuristic algorithm should be considered. There is also the aspect of the influence of the services on each other, as can be seen in equation (4). This problem has been mentioned in [18] and [19] and provides a possible path to further enriching the algorithm proposed in this paper. The resulting algorithm could minimize the interaction of services in different categories, which should positively impact cost savings for clients (services should execute with no delays). Such a solution will be considered in further research.

Provisioning problems have been tested in a TASKcloud test environment, which also confirms the presented simulation results. The next step is to implement the provisioning tool for this environment to utilize the natural possibilities of the platform.

Acknowledgments

The Regional Operational Program of the Pomeranian Voivodeship for 2014–2020, Project Number RPPM.01.02.00-22-0001 / 17, "Establishment of the Competence Center STOS (Smart and Transdisciplinary knOwledge Services) in Gdansk in the field of R&D infrastructure."

References

1. Odun-Ayo, I., Udemezue, B., Kilanko, A.: Cloud Service Level Agreements and Resource Management Advances. Science Technology and Engineering Systems Journal 4(5), 228–236 (2019).
2. Manvi, S., Shyam, G.K.: Cloud Computing Concepts and Technologies. 1st ed. CRC Press, (2021).
3. Jennings, B., Stadler, R.: Resource Management in Clouds: Survey and Research Challenges. Journal of Networks and System Management 23, 567–619 (2015).

4. Gonzalez, N., Carvalho, T., Miers, C.: Cloud resource management: towards efficient execution of large-scale scientific applications and workflows on complex infrastructures. Journal of Cloud Computing 6, (2017).
5. Orzechowski P.: Task Cloud Infrastructure in the Centre of Informatics – Tricity Academic Supercomputer & Network. TASK Quarterly 22, 313–319 (2018).
6. Beaumont, O., Eyraud-Dubois, L., Guermouche, A., Lambert, T.: Comparison of Static and Dynamic Resource Allocation Strategies for Matrix Multiplication. In: 26th IEEE International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), Florianopolis (2015).
7. Sumalatha, K., Anbarasi, M. S.: Provisioning Cloud Resources: Optimization Techniques for Resource Provisioning in Cloud Environment. 1st ed. Lambert Academic Publishing (2019).
8. Panwar, R., Supriya, M.: Dynamic resource provisioning for service-based cloud applications: A Bayesian learning approach. Journal of Parallel and Distributed Computing 168, 90–107 (2022).
9. Perumal, K., Mohan, S., Frnda, J. et al.: Dynamic resource provisioning and secured file sharing using virtualization in cloud azure. Journal of Cloud Computing 11, (2022).
10. Kashyap, D., Viradiya, J.: A survey of various load balancing algorithms in cloud computing. International Journal of Scientific & Technology Research 3(11), 115–119 (2014).
11. Mesbahi, M., Rahmani, A.: Load Balancing in Cloud Computing: A State of the Art Survey. International Journal of Modern Education and Computer Science 8, 64–78 (2016).
12. Sharma, M., Sharma, P.K., Sharma, S.S.: Efficient Load Balancing Algorithm in VM Cloud Environment. International Journal of Computer Science and Technology 3(1), 439–441 (2012).
13. Ponto, R., Kecskeméti, G., Mann, Z.: Comparison of workload consolidation algorithms for cloud data centers. Concurrency Computation Practice Experience, 1–24 (2021).
14. Ferreto, T.C., Netto, M.A., Calheiros, R.N., Rose, C.A.D.: Server consolidation with migration control for virtualized data centers. Future Generation Computer Systems 27(8), 1027–1034 (2011).
15. Armant, V., De Cauwer, M., Brown, K.N., O'Sullivan, B.: Semi-online task assignment policies for workload consolidation in cloud computing systems. Future Generation Computer Systems 82, 89–103 (2018).
16. Hasan, M., Goraya, M.S.: Fault tolerance in cloud computing environment: A systematic survey. Computers in Industry 99, 157–172 (2018).
17. Rehman, A.U., Aguiar, R., Barraca, J.P.: Fault-Tolerance in the Scope of Cloud Computing. IEEE Access 10, 63422–63441 (2022).
18. Orzechowski, P.: Complementary oriented allocation algorithm for cloud computing. TASK Quarterly 21(4), 395–403 (2017).
19. Orzechowski, P., Proficz, J., Krawczyk, H., Szymański, J.: Categorization of Cloud Workload Types with Clustering. Proceedings of the International Conference on Signal, Networks, Computing, and Systems 395, 303–313 (2016).