UDC 004.9 **doi: 10.26906/SUNZ.2024.3.196**

Yuliia Andrusenko[1], Dmytro Lysytsia[2]

[1] Kharkiv National University of Radio Electronics, Kharkiv, Ukraine
[2] National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine

IAAS PERFORMANCE ASSESSMENT WITH SERVICE LEVELS

Abstract. The paper proposes a modification of the IaaS cloud model. To demonstrate the practicality and competitiveness of the method, a comprehensive performance study is conducted using simulation. Workloads based on real production runs of heterogeneous HPC systems are used to evaluate the practicality of the scheduling method. An online scheduling problem is considered. Jobs arrive one after another, and after a new job arrives, the scheduler must decide whether to reject this incoming job or schedule it on one of the machines. The problem is online, since the scheduler must solve it without information about the next jobs.

Keywords: IAAS, service levels, distribution, jobs.

Introduction

Infrastructure as a Service (IaaS) is a cloud computing service model where computing resources are hosted in a public cloud, private cloud, or hybrid cloud. With the IaaS model, it is possible to partially or completely move an on-premises or distributed data center infrastructure to the cloud, where it is maintained and managed by a cloud provider [1–4].

The main problem of the paper is online scheduling. Jobs arrive one after another [5, 6]. When a new job arrives, the scheduler must decide whether to reject this new job or schedule it on one of the machines. The problem is online. Because the scheduler must solve it without information about the next jobs [7–9]. For this problem, the performance of the algorithms is evaluated by a set of metrics. They include the contention factor and the number of accepted jobs [10].

In the beginning, two greedy algorithms with one service level are investigated [11]. In both cases, the key properties of the service level must be met to provide benefits for real-world settings. Since the service level is often considered as a successor to the real-time paradigm, the service-oriented one with deadlines [12–14].

Basic material

Consider a set of service levels offered by an service level agreement (SLA). Let it be S [S_1, S_2, ,…, S_l, …]. For a given SL_j^l, a job j requires a throughput of s_j^l, which is guaranteed by the provisioning of the corresponding virtual machine VM, and incurs a cost Sl per unit of execution time depending on the required urgency. This urgency is expressed through the slack factor $f_j^l \geq 1$; $u_{max} = \max\{u_j^l\}$ denotes the maximum cost for all l=1…k and j =1…n. The total number of jobs submitted to the system is n_r.

Each job j is described by a tuple ‹r_j, w_j, d_j, S_j, SL_j^l› containing the release date r_j, the amount of work w_j describing the computational load that must be executed before the required response time, the deadline d_j, and the service level SL_j^l.

Let $p_j = w_j / s_j^l$ be the guaranteed time that the system will spend processing the job before the deadline, according to the service level SL_j^l. Let d_j be the latest time by which the system must execute job J_j if it is accepted. This value is calculated when a job is accepted as $d_j = r_j + f_j^l p_j$. The maximum time for a job to complete is $d_{max} = \max\{d_j^l\}$. When a job is submitted for execution, its characteristics are known.

The revenue that the system will receive for executing job J_j is calculated as $u_j^l p_j$. Once a job has been submitted, the scheduler must decide whether to accept the job or not before other jobs arrive.

In order to accept job J_j, the scheduler must ensure that some machine in the system is capable of executing it before its deadline. If accepted, later submitted jobs cannot cause job J_j not to be executed before its deadline.

Once a job is accepted, the scheduler uses some rule to generate an execution plan. The set of accepted jobs $J = [J_1, J_2, \ldots J_n]$ is a subset of the incoming jobs, where $n \leq n_r$ is the number of accepted jobs.

Consider a set of heterogeneous machines $M = [M_1, M_2, \ldots, M_m]$. Each machine M_i is described by a tuple ‹s_i, eff_i› indicating its relative processing speed s_i and its energy efficiency eff_i. At a time, only a subset of all machines can accept a job. Let $M_a(t) = [M_1, M_2, \ldots]$ be such a set of admissible machines. This set is defined for each job as the subset of available machines that can execute the job without missing deadlines and can guarantee the processing power s_j for processing. Machines whose processing speed is less than the speed guaranteed by the SLA cannot accept the job.

The value of s_i is conservatively chosen such that the speedup of all applications exceeds s_i. Thus, users receive the same guarantees regardless of which processors are used. Deadlines are calculated based on the service level and cannot be changed, and the guaranteed processing time is not violated by slower processing. C_{max} denotes the execution time of all jobs.

Next, we consider a two-level scheduling approach (Fig. 1). At the upper level, the system checks whether the job can be accepted or not using the Greedy acceptance policy. If the job is accepted, the system selects a machine from the set of admissible machines to execute it at the lower level.

A greedy acceptance policy is used at the upper level. It is based on the EDD algorithm, which prioritizes jobs according to their deadlines. When a job Jj is submitted to the system, in order to determine whether to accept or reject it, the system searches for a set of machines that can execute job Jj before its deadline, ensuring that no job in a machine is late.

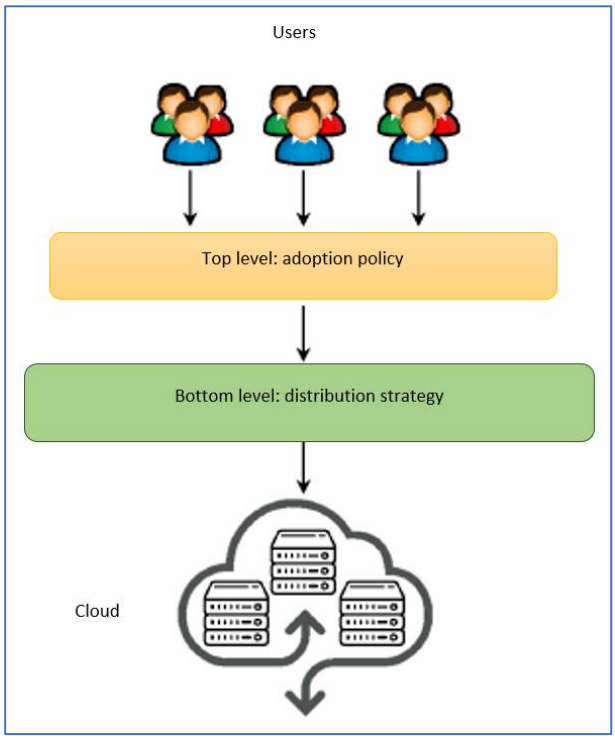

Fig. 1. Two-level scheduling approach using upper-level acceptance policy and lower-level allocation strategies

If the set of available machines is not empty ($|M_a(r_j)| \geq 1$), then job J_j is accepted, otherwise it is rejected. This completes the first stage of scheduling.

At the lower level, the Preemptive EDD algorithm is used with preemptions for each machine separately. This algorithm is easy to implement and gives an optimal solution to the problem 1 |prmp, r_j, online| L_{max}. In general, the delay L_j of job J_j is defined as (c_j – d_j, 0). Then we get $L_{max} = \max\{L_j\}$. For all plans in our problems, L_{max} must be satisfied, since none of the jobs can be late. Moreover, the Preemptive EDD algorithm creates a schedule without delays greedy scheduling and therefore does not postpone the use of resources to the future, when they may be needed by yet unknown jobs. Preemptive EDD verifies that all already accepted jobs whose due date is greater than the due date of the incoming job will be completed before their due date.

The machine for job distribution can be determined taking into account various criteria. They are characterized by the type and amount of information used to make the distribution decision.

Two levels of available information are distinguished. At level 1, it is assumed that the job execution time, machine speed and acceptance policy are known. At level 2, in addition, the machine energy efficiency and the energy consumed during the job execution are known.

Table 1 provides detailed information about the distribution strategies used in this work. The proposed strategies can be divided into three groups:

1 – without knowledge of the system, without information about jobs and resources;

2 – taking into account energy consumption, with information about energy consumption;

3 – with information about the speed of machines.

The jobs are ordered by decreasing deadlines. To execute them, jobs are taken from the head of the queue. When a new job is released, it is placed in the queue according to its deadline.

Table 1 – **Job Distribution Strategies**

Type	Strategy	Level	Description
Knowledge Free	Rand	1	Assigns work to a suitable machine chosen at random using uniform distribution over the range [1…m]
	FFit	1	Distributes work j to the first available machine capable of performing it
	MLp	1	Distributes work j to the machine with the least load at time r_j: $min\{n_i\}$
Energy aware	Max-eff	2	Distributes work to the machine with the greatest energy efficiency $\{eff_i\}$
	Min-e	2	Distributes work to the machine with the minimum total energy consumption at time r_j
	MCT-eff	2	Allocates work j to the machine with the best balance between execution time and energy efficiency, the execution time and the completion time of job k on machine i
Speed aware	Max-seff	2	Distributes work j to the machine
	Max-s	2	Distributes work j to the fastest machine: $max\{s_i\}$

EDD is an optimal algorithm for minimizing delays in a single-machine system. In our case, this corresponds to minimizing the number of rejected jobs.

Since IaaS clouds are supposed to be a promising alternative to data centers, it can be expected that the workload transferred to clouds will have similar characteristics to those transferred to real parallel and grid systems.

It is well known that the distribution of jobs is uneven in time and depends on the time of day and day of the week.

Moreover, each individual log shows a different distribution. In addition, they are recorded in different time zones.

Therefore, it is necessary to normalize the used workloads by shifting the workloads by a certain time

interval in order to represent a more realistic situation. The workloads are transformed in such a way that all traces start on the same weekday and at the same time of day.

For this purpose, all jobs until the first Monday at midnight are deleted. Time zone normalization, profiled time interval normalization, and invalid job filtering should be considered.

Conclusions

Cloud computing provides advanced computing. The paper proposes a modification of the IaaS cloud model. To demonstrate the practicality and competitiveness of the method, a comprehensive performance study is conducted using simulation.

Workloads based on real production runs of heterogeneous HPC systems are used to evaluate the practicality of the scheduling method.

An online scheduling problem is considered. Jobs arrive one after another, and after a new job arrives, the scheduler must decide whether to reject this incoming job or schedule it on one of the machines. The problem is online, since the scheduler must solve it without information about the next jobs.

REFERENCES

1. John J. Prevost, Kranthi Manoj Nagothu, Brian Kelley, and Mo Jamshidi, “Prediction of cloud data center networks loads using stochastic and neural models”, 2011 6th International Conference on System of Systems Engineering, 12138393, 2011, doi: 10.1109/SYSOSE.2011.5966610.
2. S. Kianpisheh, and R. H. Glitho, “Cost-efficient server provisioning for deadline-constrained VNFs Chains: A parallel VNF processing approach”, Proceeding of 2019 16th IEEE Annual Consumer Communications & Networking Conference, 2019. doi: 10.1109/CCNC.2019.8651799.
3. N. Kuchuk, O. Shefer, G. Cherneva, and F. A. Alnaeri, “Determining the capacity of the self-healing network segment”, Advanced Information Systems, vol. 5, no. 2, pp. 114–119, Jun. 2021, doi: 10.20998/2522-9052.2021.2.16.
4. Ye. Qiang, and W. Zhuang, “Distributed and adaptive medium access control for internet-of-things-enabled mobile networks”, IEEE Internet of Things Journal, 2017, vol. 4, no. 2, pp. 446-460, doi: 10.1109/JIOT.2016.2566659.
5. G. Kuchuk, S. Nechausov, and V. Kharchenko, “Two-stage optimization of resource allocation for hybrid cloud data store”, International Conference on Information and Digital Technologies, Zilina, 2015, pp. 266-271, doi: 10.1109/DT.2015.7222982.
6. H. Khudov, K. Tahyan, V. Chepurnyi, I. Khizhnyak, K. Romanenko, A. Nevodnichii, and O. Yakovenko, “Optimization of joint search and detection of objects in technical surveillance systems”, Advanced Information Systems, 2020, Vol. 4, No. 2, pp. 156-162, doi: 10.20998/2522-9052.2020.2.23.
7. S. Semenov, O. Sira, S. Gavrylenko, and N. Kuchuk, “Identification of the state of an object under conditions of fuzzy input data”, Eastern-European Journal of Enterprise Technologies, Vol 1, No 4 (97), pp. 22-30, 2019, doi: 10.15587/1729-4061.2019.157085.
8. S. Semenov, and Cao Weilin, “Testing process for penetration into computer systems mathematical model modification”, Advanced Information Systems, Vol. 4, No. 3, pp. 133–138. 2020, doi: 10.20998/2522-9052.2020.3.19.
9. H. Attar, M.R. Khosravi, S.S. Igorovich, K.N. Georgievan, and M. Alhihi, “E-health communication system with multiservice data traffic evaluation based on a G/G/1 analysis method”, Current Signal Transduction Therapy, 16(2), 2021, doi: 10.2174/1574362415666200224094706.
10. P. Franti, “Efficiency of random swap clustering”, Journal of Big Data, vol. 5, no. 13, 2018, pp. 1-29, doi: 10.1186/s40537-018-0122-y.
11. A. Nechausov, I. Mamusuĉ, and N. Kuchuk, “Synthesis of the air pollution level control system on the basis of hyperconvergent infrastructures”, Advanced Information Systems, vol. 1, no. 2, 2017, pp. 21–26. DOI: 10.20998/2522-9052.2017.2.04.
12. A. Kovalenko, H. Kuchuk, N. Kuchuk, and J. Kostolny, “Horizontal scaling method for a hyperconverged network”, International Conference on Information and Digital Technologies 2021, IDT 2021, pp. 331–336, 9497534, 2021, doi: https://doi.org/10.1109/IDT52577.2021.9497534.
13. H. Attar, M.R. Khosravi, S.S. Igorovich, K.N. Georgievan, and M. Alhihi, “Review and performance evaluation of FIFO, PQ, CQ, FQ, and WFQ algorithms in multimedia wireless sensor networks”, International Journal of Distributed Sensor Networks, 16(6), June 2020, doi: https://doi.org/10.1177/1550147720913233.
14. N. Kuchuk, O. Mozhaiev, S. Semenov, A. Haichenko, H. Kuchuk, S. Tiulieniev, M. Mozhaiev, V. Davydov, O. Brusakova, and Y. Gnusov, “Devising a method for balancing the load on a territorially distributed foggy environment”, Eastern-European Journal of Enterprise Technologies, vol. 1(4 (121), pp. 48–55, 2023, doi: https://doi.org/10.15587/1729-4061.2023.274177

Received (Надійшла) 19.05.2024
Accepted for publication (Прийнята до друку) 21.07.2024

Оцінка продуктивності Інтернету хмарної моделі IaaS

Ю. О. Андрусенко, Д. О. Лисиця

Анотація. У статті пропонується метод модифікації хмарної моделі IaaS. Щоб показати практичність та конкурентоспроможність методу, проведено комплексне дослідження продуктивності запропонованого методу за допомогою моделювання. Для оцінки практичності методу планування використовуються робочі навантаження, що ґрунтуються на реальних виробничих трасах гетерогенних систем. Розглядається проблема онлайн-планування. Роботи надходять одна за одною, і після надходження нової роботи планувальник повинен вирішити, чи відхилити йому поточну роботу, чи запланувати її на одній з віртуальних машин хмарної моделі IaaS. Проблема вирішується онлайн, оскільки планувальник повинен вирішувати її без інформації щодо наступних робіт.

Ключові слова: IAAS, рівні обслуговування, розподіл, роботи.