Node capacity aware load balancing algorithm for unpredictable workload distribution in cloud computing system

Xingran Ruan
Supervisor: Rizos Sakellariou

1 Background

The world is entering the Cloud computing era. Cloud computing is a new business model and service model which doesn't rely on local computer to do computing but on computing resource operated by third parties that provide computing, storage and networking resources[1]. Relying on the huge computing resource behind cloud computing, consumers are able to process big-data jobs, develop self program application and store/access large scale data with high performance. These functions are accomplished with the help of three services, Software as a service(SaaS), platform as a service(PaaS) and Infrastructure as a service(IaaS). The cloud computing running with Pay-as-you-go plan which is high cost-efficient service model[2].
The service quality is efficient if all computing resources are in balance level and this quality is achieved by resource scheduling model which manages available resource in the system. In conventional, the scheduling model plays a role in arranging tasks processing sequence and dispatching arrival tasks to available computing nodes. After the tasks and tasks' requirements are submitted to the cloud service provider, one mapping strategy between tasks and node will be generated. This strategy describes which node processes which task. Without load balancing algorithms, some node will get a heavy workload and some will get less workload at the same time while processing users' tasks. As a result, the computing node is left with imbalanced load distribution which is a negative influence on tasks computing and resource utilization. The problem of load imbalance is an unexpected condition that degrades the performance of Quality of Service(QoS) and the efficacy of the computing system. In addition, it brings the challenge on Service Level Agreement(SLA) between provider and consumer. Under these circumstances, the need for load balance algorithm research has arisen and the proposed load balancing algorithms are expected to generate a mapping strategy from tasks to nodes to avoid overload, under-load or idle running status. However, due to the arisen of requirements of using cloud service, the problem like unpredictable arriving workloads brings new challenges to existing load balancing algorithms. The unpredictable workloads may cause workload imbalance on nodes, which results in performance deterioration and violation of service level agreements (SLAs)[3], [4].

2 Definition

I display the definition of all parameters and terminology in this section.

Table 1: The parameters used in the report

Parameter	Description
N	Total number of nodes
t	Time interval
$i \in [1, 2, \cdots, N]$	Node with index i
$Q_i(t)$	Load capacity of $node_i$
$P_i(t)$	Weight of workload on $node_i$
$F_i(t)$	Average load on $node_i$
α	Preferred balance level
α_i	Load level on $node_i$
δ	Balance strictness factor
$H_i(F), i \in [1, 2, 3]$	Penalty function for average load
$C(Q)$	Cost function for cumulative penalty
λ_i	Factor of Poission distribution for $node_i$
I	Load imbalance value
θ	The difference in load level to preferred load level.

Definition 2.1. The weight of workload is unpredictable if the predicted value $\hat{P}$ satisfies:

$$\Pr(|\hat{P} - P_{true}| \geq e) > 0 \tag{1}$$

where P_{true} is the real weight of workload and e is an error threshold.

Definition 2.2. $Node_i$'s load capacity $Q_i(t)$ is the maximum number of operation the cores in the $node_i$ can run over a time period t.

Definition 2.3. The weight $P_i(t)$ is the workload allocated on $node_i$ with P_i times of operation required to complete over t time period. E.g., $node_i$ needs 1 time operation to compute the workload with $P_j = 1$ on it.

Definition 2.4. The $node_i$'s average load $F_i(t)$ over the last t period is the number of operation the node performs in t time period(it is obvious that $F_i(t) \leq Q_i(t)$). E.g., $F_i(t) = 1$ means the $node_i$ complete 1 time operation over the given t period.

Definition 2.5. The balance level $\alpha \in [0, 1]$ is a predefined parameter which describes the preferred load level in each node.

Definition 2.6. The load level $\alpha_i(t) \in [0, 1]$ over t period in $node_i$ is the percentage of the average load over the node's load capacity in the last t period.

$$\alpha_i(t) = \frac{F_i(t)}{Q_i(t)}. \tag{2}$$

Definition 2.7. The imbalance level $\delta \in [0, \min(\alpha, 1 - \alpha)]$ discribe the strictness to a given balance level α. E.g., the bigger imbalance level δ, the larger acceptable balanced interval $[\alpha - \delta, \alpha + \delta]$.

Definition 2.8. The distribution of load in nodes is balance over t period if given a predefined balance level α and imbalance level δ the average load in each $node_i$ satisfy:

$$\alpha - \delta \leq \alpha_i(t) \leq \alpha + \delta. \tag{3}$$

Definition 2.9. The performance of $node_i$ is redundancy over a time period t, if

$$\alpha_i(t) < \alpha - \delta. \tag{4}$$

Definition 2.10. The performance of $node_i$ is insufficient over a time period t, if

$$\alpha_i(t) > \alpha + \delta. \tag{5}$$

3 Motivation

Load imbalance problem always is a hot spot in the research field. To optimize the load balancing in nodes, a large body of work has been done to improve the existing algorithm's performance by taking into account different factors[5]–[35]. The cloud computing system not only processing predictable workloads but also has to process totally unpredictable arriving workloads. In those arriving workloads, some of the tasks require a large number of operations to finish, some only require a small number of operations. These unpredictable workloads bring a challenge for the task scheduler that the workload distributed on nodes may be imbalanced, some nodes are idle while others are in overload. To evenly schedule the arriving unpredictable workloads, the proposed algorithms for scheduler can generate a mapping strategy to dispatch the tasks to nodes by round-robin scheduling, power-of-d-choices, or other heuristic algorithms[6], [7], [16], [17], [27], [30], [31], [34]. And during processing the arrived workloads, the proposed load balancing algorithms move tasks from overload nodes to under-load nodes depending on local policy[5], [22], [32], [33].
However, even though the distribution of arriving unpredictable workloads is balance:

$$\alpha_i = \alpha_j, i, j \in [1, 2, \cdots, N], i \neq j.$$

problems still exist in the node system that insufficient performance ($\alpha_i > \alpha + \delta$) and redundancy performance ($\alpha_i < \alpha - \delta$). Because of the given node capacity, Q_i does not match with arriving workloads P_i. These two kinds of node status will cause bad resource waste, decrease the system performance and increase workloads' completion time. For example in Fig.1, the arriving workload $P_i(t)$ over t time period is much less than node capacity, $P_i(t) < Q_i(t)$, so the load level α_i in $node_i$ is relative low. In this case, even though the load level on nodes is even, the redundant computing resources still be wasted. Also in instant Fig.2, the arriving workload P_i is higher than node capacity, $P_i(t) > Q_i(t)$, in this case the load level $\alpha_i = 1$ all the time which is bad for device lifespan and system throughput. So it is necessary to find a feasible node capacity for the arriving unpredictable workload. From my knowledge base, present load balancing algorithms have not taken into account finding a feasible node capacity for unpredictable workloads. Thus, this project aims to fulfil this gap by calculating capacity Q_i for each node and fixes the load level α_i into a predefined interval $[\alpha - \delta, \alpha + \delta]$ when unpredictable workloads arrive.

4 Related work

Lots of work has been done to address the arriving unpredictable workloads in the distributed system[6], [16]–[18], [20], [22], [27]–[35]. In terms of solving unpredictable workload balance problem, power-of-d-choice is one interesting heuristic load balancing algorithm for solving unpredictable workloads distribution. When a set of workloads arrive, it selects d number of nodes as candidates, then allocates the workload to the node with the shortest queue[36]. An analysis of the performance of power-of-d-choices with the different number d is studied by Mitzenmacher et al. and they have proved that $d = 2$ choices leads to exponential improvements in average waiting time while the number of nodes increases to infinity[27]. Based on the work accomplished by Mitzenmacher, Ousterhout et al. improve the algorithm performance by partitioning all unpredictable workloads into a number of small chunks when considerable unpredictable workloads arrive[30]. Ying et al. further improve the load balancing performance with less number of candidates selected in the power-of-d-choices algorithm when distributing considerable unpredictable workloads[34]. Tang et al. and Zhang et al. take into account the features of unpredictable workloads when applying power-of-d-choice to allocate the arriving tasks evenly[28], [29]. However, those improvements of power-of-d-choices ignore discussing the value of nodes' capacity, thus the balanced level may be low in a high capacity system or be high in a low capacity system.

Round-robin is another common load balancing algorithm to solve unpredictable workload distribution. In this scheme, the arriving workloads are dispatched to different nodes in a round-robin manner. But this static algorithm does not consider the current load of the node. To fix this disadvantage, Xu et al. consider the load of the next candidate and skip the present round if the load on the next node is the biggest when a new unpredictable workload arrives[37]. Patel et al. propose a modified Round-robin scheduling by taking into account the task's priority for those unpredictable workloads[38]. However, the improvement does not study the feasible capacity on the node, thus the performance of the algorithm on solving unpredictable workloads still can be improved.

Besides generating a load balancing strategy for arriving unpredictable workloads, moving the workloads from overload nodes to under-load nodes during processing the workloads is an efficient solution for unpredictable workload distribution. Wang et al. randomly distribute the unpredictable workloads on nodes at first and then proposing a task stealing algorithm to steal tasks from overload node's queue to under-load node's queue[33]. Based on the previous work, Wang et al. consider the data-intensive workloads while balancing the workloads on nodes. The improved algorithm adds a data size tag on each arriving unpredictable load to avoid transfer those tasks with large size data sets[32]. Oncioiu et al., improve the algorithm's performance[33] in an overwhelmingly large size of information scenario[5]. In the proposed algorithm, the nodes are classified into three class, running nodes, idle nodes and dispatcher nodes. The arriving unpredictable workloads are distributed on idle nodes by Round-robin scheduling then the load balancer moving workloads from the queue of running nodes to idle nodes in its neighbour. A similar heuristic algorithm is proposed in [22]. The algorithm first sorts the running node with increasing order by load level then moving those unpredictable workloads from overload node to the last few nodes in the list. However, the algorithms relate with re-migrating load procedure still ignore the importance of configuring a feasible node's capacity for unpredictable workloads.

5 Problem Description

The node's average load is a measure of the amount of computational task that physical machines deployed on node perform over a period of time t. There are N nodes in the cloud computing system which is represented by index $i = [1, 2, \cdots, N]$. Each node contains its load capacity over t time period $Q_i(t)$. And the totally weight of workload allocated on $node_i$ over t time period is $P_i(t)$. The problem that I consider is defined as follow:

Input:

The number of nodes N.

The time period t.

The expected arriving workload λ_i.

Preferred balance level α.

Imbalance level δ.

Output:

Node capacity vector $(Q_1(t), Q_2(t), \cdots Q_N(t))$.

Assuming the total weight of unpredictable workloads P_i in unit time Δt applies Poisson distribution with factor λ_i, $P_i \sim Poi(\lambda_i)$, where λ_i is the expectation of arriving weight on $node_i$ at unit time Δt. Given a preferred balance level α and a balance strictness factor δ, propose an algorithm giving penalty feedback on performance redundant status and insufficient performance status to compute the feasible value $(Q_1(t), Q_2(t), \cdots Q_N(t))$ to minimize the difference between α_i and α in the case of balancing unpredictable workloads.

6 Objectives

The main objectives of my PhD thesis are as follows:

- Improve load balancing algorithm by calculating node load capacity $Q_i, i \in [1, 2, \cdots, N]$ that can make the load level $\alpha_i, i \in [1, 2, \cdots, N]$ in the fixed interval $[\alpha - \delta, \alpha + \delta]$ when unpredictable workloads arrive.
- Extend the algorithm by considering other key factors in a different scenario.
- Complete a theory about maintaining load balancing for unpredictable workloads by controlling the node's capacity.
  (a) Study the factors that influence the performance of the node capacity aware load balancing algorithm in the case of unpredictable workloads.
  (b) The existance of $Q_i, i \in [1, 2, \cdots, N]$ for the arriving unpredictable workloads.
  (c) The sensitive analysis of $Q_i, i \in [1, 2, \cdots, N]$ to $\alpha_i, i \in [1, 2, \cdots, N]$.

7 Contribution

This project makes the following new contributions that extend the previous work broader in scope and more in-depth:

- Devise an analytical model to analyze the node capacity for balancing unpredictable workloads.
- Improve the exist the performance of load balance algorithms when unpredictable workloads arriving and fix the load level on the node in a predefined interval.
- Provide a theoretical foundation to configure the node system.

8 Metric

To evaluate the proposed algorithms solving this problem, metrics are necessary.

- Load imbalance I: The average load difference in nodes over a time period T. The less the imbalance value, the better the algorithm performance.

$$I = 1 - \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \|load(Node(i)) - load(Node(j))\|}{(N-1) \sum_{i=1}^{N} load(Node(i))} \tag{6}$$

- Load capacity: The capacity on each node over a time period t. Compare with average value $\bar{Q(t)}$ generated from different algorithms, the lower $\bar{Q(t)}$ the better algorithm performance. Because the lower $\bar{Q(t)}$ can handle same weight of workload in balance level with less cost.

$$\bar{Q(t)} = \sum_{i=1}^{N} Q_i(t)/N \tag{7}$$

- Load level: The performance insufficient exist when load level on nodes is relatively high and the redundant performance exist when load level on nodes is relatively small. So algorithm performance better if the load level on nodes close to the preferred load level.

$$\theta(t) = \sum_{i=1}^{N} \|\alpha - \alpha_i\|/N \tag{8}$$

9 Methodology

Studying this problem aims to fulfil the gap of load balancing in cloud computing. In the defined scenario, some unpredicted tasks in are arriving at $node_i$ with weight $P_i(t)$ in total over an interval t. The goal is finding the optimal node's capacity $Q_i(t)$ which controls the balanced load level α_i in each node in a predefined balance level interval $[\alpha - \delta, \alpha + \delta]$. This procedure displays in Fig.3
Given a function $f_i = f_i(Q, P, t), i \in [1, 2, \cdots, N]$ for the load of finished processed

tasks in $node_i$, and a penalty function $H(F_i)$ for different load level F_i on $node_i$, where $F_i = F_i(P_i, f_i, t)$ is the average load in the node.

$$H(F) = \begin{cases} H_1(F) & F \leq Q' - \delta Q, \\ H_2(F) & Q' - \delta Q < F < Q' + \delta Q, \\ H_3(F) & Q' + \delta Q \leq F \leq Q \end{cases} \tag{9}$$

where $Q' = \alpha Q_i$ is preferred average load. In the equation eq.9, $H_1(F)$ represents the penalty for the low load level in node. $H_2(F)$ represents the positive effect for a balanced load level. $H_3(F)$ represents the penalty for the overload level in node. In this way, we obtain a cost function C_i for load status in $node_i$,

$$C_i = \mathbb{E}\, H(F_i) = \int_0^{\infty} H(F_i) dP_i \tag{10}$$

Because the relation of Q_i and C_i in 10 is not explicit and the limits of integration are 0 and ∞, I change the measure from P_i to measure F_i by

$$dP_i = f(P_i) dF_i \tag{11}$$

Then the equation eq.10 can be transfered into

$$\begin{aligned} C_i(Q) &= \int_0^{Q_i} H(F_i) f(P_i) dF_i \\ &= \int_0^{Q' - \delta Q_i} H_1(F_i) f(P_i) dF_i + \int_{Q' - \delta Q_i}^{Q' + \delta Q_i} H_2(F_i) f(P_i) dF_i + \int_{Q' + \delta Q_i}^{Q_i} H_3(F_i) f(P_i) dF_i. \end{aligned} \tag{12}$$

We prefer to control load level $\alpha_i, i \in [1, 2, \cdots, N]$ in the interval $[\alpha - \delta, \alpha + \delta]$, thus we maximize $\|C(Q)\|_2$ by finding the vector Q. It can be done by calculating $\frac{dC_i}{dQ_i}$.

10 Performance evaluation

In this section, I will propose my experiment plan to assess the method in Section9. To allow repeatable evaluation, I will develop a simulation environment in CloudSim that allow me to compare the performances of different algorithms using unpredictable workload sets from the real application or a public resource.

I will collect a set of requests in t time period from a real application or one public resource and set up a default configuration(node capacity, number of nodes) to the node system.

I consider the load level on nodes in the case of applying power-of-d-choices, round-robin and other state-of-art algorithms to distribute the unpredictable workloads. I will calculate α_i on each node.

After finishing computing the load imbalance I, load capacity $\bar{Q(t)}$ and load level $\theta(t)$ of different algorithms in the default node system(without feasible node capacity). I will configure a new node capacity setting by the method in Section9 and apply power-of-d-choices, round-robin and other state-of-art algorithms to distribute workloads on the new node system. To evaluate the improvement, I will compute load imbalance I, load capacity $\bar{Q(t)}$ and load level $\theta(t)$ in the new system.

In the end, analyze the value of the metrics from the experiment.

11 Research plans

In order to achieve the research objectives mentioned in this report, we propose a detailed plan in the next two years listed below:

- **Plans for objective 1**
  - **1.07.2020-31.08.2020**
    Complete the method in Section9, including explicit formula of $f_i(Q, P, t)$,$H(F)$ and $F_i(P_i, f_i, t)$, and the existance of $f(P_i)$ in eq.11.
  - **1.09.2020-30.09.2020**
    Complete numerical analysis of the algorithm in Section9.
  - **1.10.2020-31.10.2020**
    Improve the state-of-art algorithms by taking into account node capacity and analyze the improvements by metrics.
  - **1.11.2020-31.12.2020**
    Write a report based on the developed model and numerical results.
- **Plans for objective 2**
  - **1.01.2021-31.03.2021**
    Extend the algorithm by considering factor transfer time duration in a geo-distributed system then comparing the performance with state-of-art algorithms.
  - **1.04.2021-31.05.2021**
    Design an experiment to evaluate the improvement of the extension and write a report based on the model and numerical results.
  - **1.06.2021-30.06.2021**
    Write a progress report about finished work for second year report.
  - **1.07.2021-30.09.2021**
    Extend the algorithm by considering dependent arriving unpredictable workloads then comparing the performance with state-of-art algorithms.
  - **1.10.2021-30.11.2021**
    Design an experiment to evaluate the improvement of the extension and write a report based on the model and numerical results.
- **Plans for objective 3**
  - **1.12.2021-31.12.2021**
    Analyze the factor that affect the performance of proposed algorithms.
  - **1.01.2022-31.03.2022**
    Complete the sensitive analysis of Q_i to α_i in case of arriving workloads are totally unpredictable.
  - **1.04.2022-31.08.2022**
    Summarize the work of last three years and write PhD thesis.

References

[1] W. D. Tian and Y. D. Zhao, *Optimized cloud resource management and scheduling: theories and practices*. Morgan Kaufmann, 2014.

[2] W. Venters and E. A. Whitley, "A critical review of cloud computing: Researching desires and realities," *Journal of Information Technology*, vol. 27, no. 3, pp. 179–197, 2012.

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

[4] J. O. Gutierrez-Garcia and A. Ramirez-Nafarrate, "Agent-based load balancing in cloud data centers," *Cluster Computing*, vol. 18, no. 3, pp. 1041–1062, 2015.

[5] A.-R. Oncioiu, F. Pop, and C. Esposito, "Asymptotic load balancing algorithm for many task scheduling," in *International Conference on Ad-Hoc Networks and Wireless*, Springer, 2019, pp. 136–149.

[6] J Prassanna and N. Venkataraman, "Threshold based multi-objective memetic optimized round robin scheduling for resource efficient load balancing in cloud," *Mobile Networks and Applications*, vol. 24, no. 4, pp. 1214–1225, 2019.

[7] E. C. P. Neto, G. Callou, and F. Aires, "An algorithm to optimise the load distribution of fog environments," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2017, pp. 1292–1297.

[8] H.-Y. Kim and J.-M. Kim, "A load balancing scheme based on deep-learning in iot," *Cluster Computing*, vol. 20, no. 1, pp. 873–878, 2017.

[9] S. Riaz and U. Park, "Power control for interference mitigation by evolutionary game theory in uplink noma for 5g networks," *Journal of the Chinese Institute of Engineers*, vol. 41, no. 1, pp. 18–25, 2018.

[10] D.-g. Zhang, C. Chen, Y.-y. Cui, and T. Zhang, "New method of energy efficient subcarrier allocation based on evolutionary game theory," *Mobile Networks and Applications*, pp. 1–14, 2018.

[11] C. H. Tseng, "Multipath load balancing routing for internet of things," *Journal of Sensors*, vol. 2016, 2016.

[12] X. Wang, M.-J. Sheng, Y.-Y. Lou, Y.-Y. Shih, and M. Chiang, "Internet of things session management over lte—balancing signal load, power, and delay," *IEEE Internet of Things Journal*, vol. 3, no. 3, pp. 339–353, 2015.

[13] S. Rashidi and S. Sharifian, "A hybrid heuristic queue based algorithm for task assignment in mobile cloud," *Future Generation Computer Systems*, vol. 68, pp. 331–345, 2017.

[14] L. Liu, Y. Du, and Q. Fan, "A constrained multi-objective computation offloading algorithm in the mobile cloud computing environment.," *KSII Transactions on Internet & Information Systems*, vol. 13, no. 9, 2019.

[15] C. Li, J. Tang, T. Ma, X. Yang, and Y. Luo, "Load balance based workflow job scheduling algorithm in distributed cloud," *Journal of Network and Computer Applications*, vol. 152, p. 102 518, 2020.

[16] M. N. Arab and M. Sharifi, "A model for communication between resource discovery and load balancing units in computing environments," *The Journal of Supercomputing*, vol. 68, no. 3, pp. 1538–1555, 2014.

[17] Y. Liu, C. Zhang, B. Li, and J. Niu, "Dems: A hybrid scheme of task scheduling and load balancing in computing clusters," *Journal of Network and Computer Applications*, vol. 83, pp. 213–220, 2017.

[18] K.-M. Cho, P.-W. Tsai, C.-W. Tsai, and C.-S. Yang, "A hybrid meta-heuristic algorithm for vm scheduling with load balancing in cloud computing," *Neural Computing and Applications*, vol. 26, no. 6, pp. 1297–1309, 2015.

[19] R. K. Naha and M. Othman, "Cost-aware service brokering and performance sentient load balancing algorithms in the cloud," *Journal of Network and Computer Applications*, vol. 75, pp. 47–57, 2016.

[20] S.-L. Chen, Y.-Y. Chen, and S.-H. Kuo, "Clb: A novel load balancing architecture and algorithm for cloud services," *Computers & Electrical Engineering*, vol. 58, pp. 154–160, 2017.

[21] Y. Mao, D. Ren, and X. Chen, "Adaptive load balancing algorithm based on prediction model in cloud computing," in *Proceedings of the Second International Conference on Innovative Computing and Cloud Computing*, 2013, pp. 165–170.

[22] E. Y. Daraghmi and S.-M. Yuan, "A small world based overlay network for improving dynamic load-balancing," *Journal of Systems and Software*, vol. 107, pp. 187–203, 2015.

[23] J. Zhao, K. Yang, X. Wei, Y. Ding, L. Hu, and G. Xu, "A heuristic clustering-based task deployment approach for load balancing using bayes theorem in cloud environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 305–316, 2015.

[24] B. Kang and H. Choo, "A cluster-based decentralized job dispatching for the large-scale cloud," *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, no. 1, p. 25, 2016.

[25] F. Zegrari, A. Idrissi, and H. Rehioui, "Resource allocation with efficient load balancing in cloud environment," in *Proceedings of the International Conference on Big Data and Advanced Wireless Technologies*, 2016, pp. 1–7.

[26] M. M. Khalill, A. D. Khomonenko, and S. I. Gindin, "Load balancing cloud computing with web-interface using multi-channel queuing systems with warming up and cooling," in *International Symposium on Intelligent and Distributed Computing*, Springer, 2019, pp. 385–393.

[27] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 10, pp. 1094–1104, 2001.

[28] L. Tang, Z. Li, P. Ren, J. Pan, Z. Lu, J. Su, and Z. Meng, "Online and offline based load balance algorithm in cloud computing," *Knowledge-Based Systems*, vol. 138, pp. 91–104, 2017.

[29] Z. Y. Zhang and M. Moh, "Randomized load balancing for cloud computing using bacterial foraging optimization," in *Proceedings of the 2019 ACM Southeast Conference*, 2019, pp. 117–124.

[30] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica, "Batch sampling: Low overhead scheduling for sub-second prallel jobs," *University of California, Berkeley*, 2012.

[31] X. Liu and L. Ying, "On achieving zero delay with power-of-d-choices load balancing," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, IEEE, 2018, pp. 297–305.

[32] K. Wang, K. Qiao, I. Sadooghi, X. Zhou, T. Li, M. Lang, and I. Raicu, "Load-balanced and locality-aware scheduling for data-intensive workloads at extreme scales," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 1, pp. 70–94, 2016.

[33] K. Wang, K. Brandstatter, and I. Raicu, "Simmatrix: Simulator for many-task computing execution fabric at exascale.," in *SpringSim (HPC)*, Citeseer, 2013, p. 9.

[34] L. Ying, R. Srikant, and X. Kang, "The power of slightly more than one sample in randomized load balancing," *Mathematics of Operations Research*, vol. 42, no. 3, pp. 692–722, 2017.

[35] H. Yan, X. Zhu, H. Chen, H. Guo, W. Zhou, and W. Bao, "Deft: Dynamic fault-tolerant elastic scheduling for tasks with uncertain runtime in cloud," *Information Sciences*, vol. 477, pp. 30–46, 2019.

[36] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal, "Balanced allocations," *SIAM journal on computing*, vol. 29, no. 1, pp. 180–200, 1999.

[37] Z. Xu and X. Wang, "A modified round-robin load-balancing algorithm for cluster-based web servers," in *Proceedings of the 33rd Chinese Control Conference*, IEEE, 2014, pp. 3580–3584.

[38] S. Patel and M. Bhatt, "Implementation of load balancing in cloud computing through round robin & priority using cloudsim," *International Journal for Rapid Research in Engineering Technology & Applied Science*, vol. 3, no. 11, 2017.

[39] M. Ghobaei-Arani, S. Jabbehdari, and M. A. Pourmina, "An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach," *Future Generation Computer Systems*, vol. 78, pp. 191–210, 2018.

[40] S. K. Battula, S. Garg, R. K. Naha, P. Thulasiraman, and R. Thulasiram, "A micro-level compensation-based cost model for resource allocation in a fog environment," *Sensors*, vol. 19, no. 13, p. 2954, 2019.

[41] M. S. Aslanpour, M. Ghobaei-Arani, and A. N. Toosi, "Auto-scaling web applications in clouds: A cost-aware approach," *Journal of Network and Computer Applications*, vol. 95, pp. 26–41, 2017.

[42] A. N. Toosi, R. O. Sinnott, and R. Buyya, "Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using aneka," *Future Generation Computer Systems*, vol. 79, pp. 765–775, 2018.

[43] T. H. Nguyen, M. Di Francesco, and A. Yla-Jaaski, "Virtual machine consolidation with multiple usage prediction for energy-efficient cloud data centers," *IEEE Transactions on Services Computing*, 2017.

[44] Z. Zhou, J. Abawajy, M. Chowdhury, Z. Hu, K. Li, H. Cheng, A. A. Alelaiwi, and F. Li, "Minimizing sla violation and power consumption in cloud data centers using adaptive energy-aware algorithms," *Future Generation Computer Systems*, vol. 86, pp. 836–850, 2018.

[45] L. Grange, G. Da Costa, and P. Stolf, "Green it scheduling for data center powered with renewable energy," *Future Generation Computer Systems*, vol. 86, pp. 99–120, 2018.

[46] Y. Li, X. Wang, P. Luo, and Q. Pan, "Thermal-aware hybrid workload management in a green datacenter towards renewable energy utilization," *Energies*, vol. 12, no. 8, p. 1494, 2019.

[47] S. MirhoseiniNejad, H. Moazamigoodarzi, G. Badawy, and D. G. Down, "Joint data center cooling and workload management: A thermal-aware approach," *Future Generation Computer Systems*, vol. 104, pp. 174–186, 2020.

[48] Z. Ding, Y. Cao, L. Xie, Y. Lu, and P. Wang, "Integrated stochastic energy management for data center microgrid considering waste heat recovery," *IEEE Transactions on Industry Applications*, vol. 55, no. 3, pp. 2198–2207, 2019.

[49] P. G. Naranjo, Z. Pooranian, S. Shamshirband, J. H. Abawajy, and M. Conti, "Fog over virtualized iot: New opportunity for context-aware networked applications and a case study," *Applied Sciences*, vol. 7, no. 12, p. 1325, 2017.

[50] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE internet of things journal*, vol. 3, no. 6, pp. 1171–1181, 2016.

[51] L. Li, M. Guo, L. Ma, H. Mao, and Q. Guan, "Online workload allocation via fog-fog-cloud cooperation to reduce iot task service delay," *Sensors*, vol. 19, no. 18, p. 3830, 2019.

[52] Q. Zhang, L. T. Yang, Z. Yan, Z. Chen, and P. Li, "An efficient deep learning model to predict cloud workload for industry informatics," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3170–3178, 2018.

[53] J. Chen, K. Li, Q. Deng, K. Li, and S. Y. Philip, "Distributed deep learning model for intelligent video surveillance systems with edge computing," *IEEE Transactions on Industrial Informatics*, 2019.

[54] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6774–6785, 2019.

[55] K. Jansen and D. Trystram, "Scheduling parallel jobs on heterogeneous platforms," *Electronic Notes in Discrete Mathematics*, vol. 55, pp. 9–12, 2016.

[56] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog computing: Towards minimizing delay in the internet of things," in *2017 IEEE international conference on edge computing (EDGE)*, IEEE, 2017, pp. 17–24.

[57] V. Sundararaj, "Optimal task assignment in mobile cloud computing by queue based ant-bee algorithm," *Wireless Personal Communications*, vol. 104, no. 1, pp. 173–197, 2019.

[58] W.-J. Wang, Y.-S. Chang, W.-T. Lo, and Y.-K. Lee, "Adaptive scheduling for parallel tasks with qos satisfaction for hybrid cloud environments," *The Journal of Supercomputing*, vol. 66, no. 2, pp. 783–811, 2013.

[59] N. Daneshfar, N. Pappas, V. Polishchuk, and V. Angelakis, "Service allocation in a mobile fog infrastructure under availability and qos constraints," in *2018 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2018, pp. 1–6.

[60] T. Lambert and R. Sakellariou, "Allocation of publisher/subscriber data links on a set of virtual machines," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, IEEE, 2018, pp. 516–523.

[61] L. Li, K. Ota, and M. Dong, "Deep learning for smart industry: Efficient manufacture inspection system with fog computing," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4665–4673, 2018.

[62] X. Xiang, W. Liu, T. Wang, M. Xie, X. Li, H. Song, A. Liu, and G. Zhang, "Delay and energy-efficient data collection scheme-based matrix filling theory for dynamic traffic iot," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, p. 168, 2019.

[63] Y. Wang, X. Wu, and H. Haas, "Distributed load balancing for internet of things by using li-fi and rf hybrid network," in *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, IEEE, 2015, pp. 1289–1294.

[64] J. W. Shin, J. S. Kim, M. Y. Chung, and S. J. Lee, "Control channel load balancing in narrow band cellular iot systems supporting coverage class," in *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, IEEE, 2016, pp. 343–348.

[65] S. J. Jazebi and A. Ghaffari, "Risa: Routing scheme for internet of things using shuffled frog leaping optimization algorithm," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–11, 2020.

[66] P. H. Barros, I. Cardoso-Pereira, L. Foschini, A. Corradi, and H. S. Ramos, "Load balancing in d2d networks using reinforcement learning," in *2019 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, 2019, pp. 1–6.

[67] A. Attiah, M. F. Amjad, M. Chatterjee, and C. Zou, "An evolutionary routing game for energy balance in wireless sensor networks," *Computer Networks*, vol. 138, pp. 31–43, 2018.

[68] V. J. Kotagi, F. Singh, and C. S. R. Murthy, "Adaptive load balanced routing in heterogeneous iot networks," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, 2017, pp. 589–594.

[69] Z. Liu, J. Li, Y. Wang, X. Li, and S. Chen, "Hgl: A hybrid global-local load balancing routing scheme for the internet of things through satellite networks," *International Journal of Distributed Sensor Networks*, vol. 13, no. 3, p. 1 550 147 717 692 586, 2017.

[70] X. Sun and N. Ansari, "Traffic load balancing among brokers at the iot application layer," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 489–502, 2017.

[71] P. G. V. Naranjo, Z. Pooranian, M. Shojafar, M. Conti, and R. Buyya, "Focan: A fog-supported smart city network architecture for management of applications in the internet of everything environments," *Journal of Parallel and Distributed Computing*, vol. 132, pp. 274–283, 2019.

[72] C. N. Le Tan, C. Klein, and E. Elmroth, "Location-aware load prediction in edge data centers," in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, IEEE, 2017, pp. 25–31.

[73] J. J. Prevost, K. Nagothu, B. Kelley, and M. Jamshidi, "Prediction of cloud data center networks loads using stochastic and neural models," in *2011 6th International Conference on System of Systems Engineering*, IEEE, 2011, pp. 276–281.

[74] Q. Guo, R. Huo, H. Meng, E Xinhua, J. Liu, T. Huang, and Y. Liu, "Research on lstm-based load prediction for edge data centers," in *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, IEEE, 2018, pp. 1825–1829.

[75] W. Fang, Z. Lu, J. Wu, and Z. Cao, "Rpps: A novel resource prediction and provisioning scheme in cloud data center," in *2012 IEEE Ninth International Conference on Services Computing*, IEEE, 2012, pp. 609–616.

[76] A. Gandhi, P. Dube, A. Karve, A. Kochut, and L. Zhang, "Adaptive, model-driven autoscaling for cloud applications," in *11th International Conference on Autonomic Computing ({ICAC} 14)*, 2014, pp. 57–64.

[77] P. Jamshidi, A. M. Sharifloo, C. Pahl, A. Metzger, and G. Estrada, "Self-learning cloud controllers: Fuzzy q-learning for knowledge evolution," in *2015 International Conference on Cloud and Autonomic Computing*, IEEE, 2015, pp. 208–211.

[78] R. Karim, C. Ding, and A. Miri, "End-to-end performance prediction for selecting cloud services solutions," in *2015 IEEE Symposium on Service-Oriented System Engineering*, IEEE, 2015, pp. 69–77.

[79] F. Nawaz, M. R. Asadabadi, N. K. Janjua, O. K. Hussain, E. Chang, and M. Saberi, "An mcdm method for cloud service selection using a markov chain and the best-worst method," *Knowledge-Based Systems*, vol. 159, pp. 120–131, 2018.

12 Appendix

Table 2: Collection of papers

Study	Objective	Workload	Method
Yan et al.[35]	Increase system fault-tolerant	Unpredictable workload	Integer programming
Ghobaei et al.[39]	Maximize resource usage	Predictable workload	Reinforce learning
Battula et al.[40]	Maximize resource usage	Predictable workload	Integer programming
Aslanpour et al.[41]	Minimize the executing cost	Predictable workload	Integer programming
Toosi et al.[42]	Minimize the executing cost	Predictable workload	Integer programming
Nguyen et al.[43]	Minimize the executing cost	Predictable workload	Self-defined heuristic algorithm
Zhou et al.[44]	Minimize the executing cost	Predictable workload	Integer programming
Grange et al.[45]	Minimize the executing cost	Predictable workload	Integer programming
Li et al.[46]	Minimize the executing cost	Predictable workload	Integer programming
Mirhoseininejad et al.[47]	Minimize the executing cost	Predictable workload	Integer programming
Ding et al.[48]	Minimize the executing cost	Predictable workload	Integer programming
Naranjo et al.[49]	Minimize the executing cost	Predictable workload	Integer programming
Deng et al.[50]	Minimize the executing cost	Predictable workload	Integer programming
Li et al.[51]	Minimize the workload completion time	Predictable workload	Integer programming
Zhang et al.[52]	Minimize the workload completion time	Predictable workload	Polyadic decomposition

Continued on next page

Table 2 – continued from previous page

Study	Objective	Workload	Method
Chen et al.[53]	Minimize the workload completion time	Predictable workload	Self-defined heuristic algorithm
Kuang et al.[54]	Minimize the workload completion time	Predictable workload	Self-defined heuristic algorithm
Jansen[55]	Minimize the workload completion time	Predictable workload	Integer programming
Yousefpour et al.[56]	Minimize the workload completion time	Predictable workload	Integer programming
Sundararaj et al.[57]	Minimize the workload completion time	Predictable workload	Queue theory
Wang et al.[58]	Minimize the workload completion time	Predictable workload	Integer programming
Daneshfar et al.[59]	Minimize the workload transfer time	Predictable workload	Integer programming
Lambert et al.[60]	Minimize the workload transfer time	Predictable workload	Integer programming
Li et al.[61]	Minimize the workload transfer time	Predictable workload	Integer programming
Xiang et al.[62]	Minimize the workload transfer time	Predictable workload	Self-defined heuristic algorithm
Wang et al.[63]	Minimize the workload transfer time	Predictable workload	Integer programming
Shin et al.[64]	Minimize the workload transfer time	Predictable workload	Self-defined heuristic algorithm
Jazebi et al.[65]	Minimize the workload transfer time	Predictable workload	Self-defined heuristic algorithm
Barros et al.[66]	Optimise the load balancing in network	Predictable workload	Reinforce learning
Attiah et al.[67]	Optimise the load balancing in network	Predictable workload	Game theory
Kotaqi et al.[68]	Optimise the load balancing in network	Predictable workload	Integer programming
Liu et al.[69]	Optimise the load balancing in network	Predictable workload	Integer programming
Sun et al.[70]	Optimise the load balancing in network	Predictable workload	Integer programming
Naranjo et al.[71]	Optimise the load balancing in network	Predictable workload	Self-defined heuristic algorithm
Neto et al.[7]	Optimise the load balancing in nodes	Predictable workload	Self-defined heuristic algorithm
Kim et al.[8]	Optimise the load balancing in nodes	Predictable workload	Deep learning model
Riaz et al.[9]	Optimise the load balancing in nodes	Predictable workload	Game theory
Zhang et al.[10]	Optimise the load balancing in nodes	Predictable workload	Game theory
Tseng et al.[11]	Optimise the load balancing in nodes	Predictable workload	Self-defined heuristic algorithm
Wang et al.[12]	Optimise the load balancing in nodes	Predictable workload	Integer programming
Rashidi et al.[13]	Optimise the load balancing in nodes	Predictable workload	Queue theory
Liu et al.[14]	Optimise the load balancing in nodes	Predictable workload	Integer programming
Li et al.[15]	Optimise the load balancing in nodes	Predictable workload	Queue theory
Arab et al.[16]	Optimise the load balancing in nodes	Unpredictable workload	Self-defined heuristic algorithm
Liu et al.[17]	Optimise the load balancing in nodes	Unpredictable workload	Self-defined heuristic algorithm
Cho et al.[18]	Optimise the load balancing in nodes	Unpredictable workload	Ant Colony with Particle Swarm
Naha et al.[19]	Optimise the load balancing in nodes	Predictable workload	Self-defined heuristic algorithm
Chen et al.[20]	Optimise the load balancing in nodes	Unpredictable workload	Self-defined heuristic algorithm
Mao et al.[21]	Optimise the load balancing in nodes	Predictable workload	Self-defined heuristic algorithm
Daraghmi et al.[22]	Optimise the load balancing in nodes	Unpredictable workload	Self-defined heuristic algorithm
Zhao et al.[23]	Optimise the load balancing in nodes	Predictable workload	Bayes theorem
Kang et al.[24]	Optimise the load balancing in nodes	Predictable workload	Self-defined heuristic algorithm
Zegrari et al.[25]	Optimise the load balancing in nodes	Predictable workload	Self-defined heuristic algorithm
Khalill et al.[26]	Optimise the load balancing in nodes	Predictable workload	Queue theory
Mitzenmacher et al.[27]	Optimise the load balancing in nodes	Unpredictable workload	Power-of-d-choice
Tang et al.[28]	Optimise the load balancing in nodes	Unpredictable workload	Self-defined heuristic algorithm
Zhang et al.[29]	Optimise the load balancing in nodes	Unpredictable workload	Self-defined heuristic algorithm
Ousterhout et al.[30]	Optimise the load balancing in nodes	Unpredictable workload	Power-of-d-choice
Liu et al.[17]	Optimise the load balancing in nodes	Unpredictable workload	Self-defined heuristic algorithm
Liu et al.[31]	Optimise the load balancing in nodes	Unpredictable workload	Power-of-d-choice
Wang et al.[32]	Optimise the load balancing in nodes	Unpredictable workload	Self-defined heuristic algorithm
Wang et al.[33]	Optimise the load balancing in nodes	Unpredictable workload	Self-defined heuristic algorithm
Ying et al.[34]	Optimise the load balancing in nodes	Unpredictable workload	Self-defined heuristic algorithm
Le et al.[72]	Predict workload on nodes	Predictable workload	Vector AutoRegression
Prevost et al.[73]	Predict workload on nodes	Predictable workload	Neural network and Wiener filter
Guo et al.[74]	Predict workload on nodes	Predictable workload	Recurrent neural network
Fang et al.[75]	Scalable computing resource scheme	Predictable workload	ARIMA
Gandhi et al.[76]	Scalable computing resource scheme	Predictable workload	Queue theory and Kalmon filter
Jamshidi et al.[77]	Scalable computing resource scheme	Predictable workload	Reinforce learning
Karim et al.[78]	Select suitable cloud services for costomer	Predictable workload	Integer programming
Nawaz et al.[79]	Select suitable cloud services for costomer	Predictable workload	Integer programming

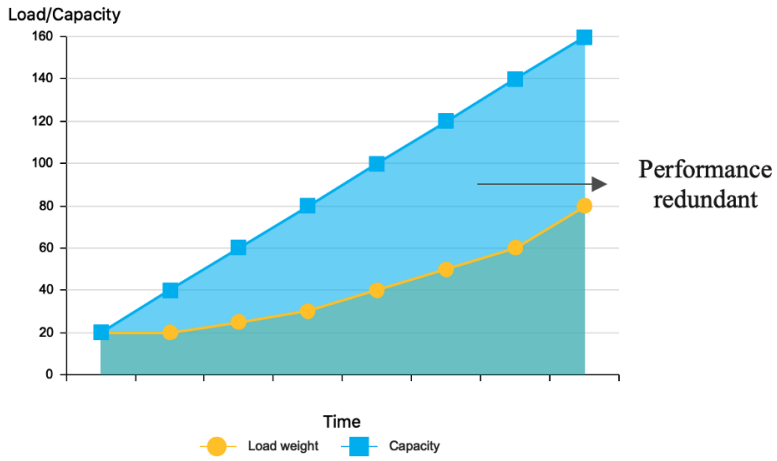

Figure 1: Performance redundant

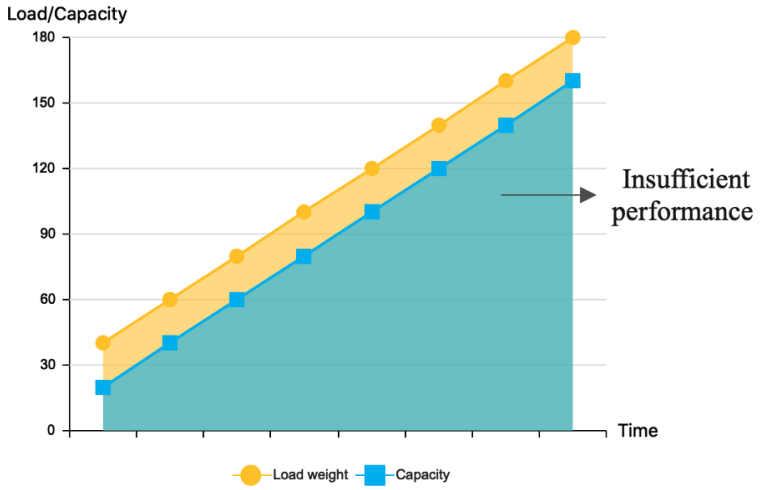

Figure 2: Inefficient performance

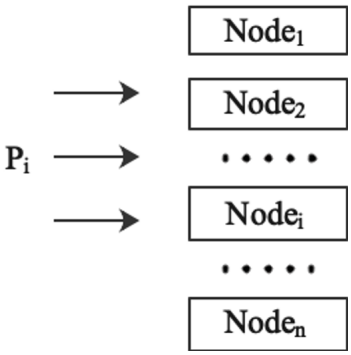

Figure 3: The processing flow in nodes