Cloud Morphing: A Responsive Model for Distributed Resource Management

Anonymous Author(s)

ABSTRACT

In this paper, we envision future distributed cloud services with abundant heterogeneous computing resources, and propose to apply dynamic pricing to a responsive model for distributed resource management. In this model, the pseudo cost of a resource is a function of the load, and the required resources for a job are allocated by a simple cost minimization algorithm. We introduce a convex cost function that enables autonomous load-balancing and idle-resource pooling within the cloud, and allows stakeholders to loosely control the utilization of each resource by manipulating the cost function or the resource weights of micro-jobs. We examine different cost tuning methods for adjusting utilization, and illustrate their behaviors by simulation.

1 INTRODUCTION

1.1 Background

Cloud computing was brought about by virtualization technologies, enabling to build large pools of virtualized computing resources (e.g., servers, storage, and network) and dynamically allocate necessary resources for providing services. However, cloud computing system itself is still on physical cloud infrastructure; cloud system operators manually manage physical servers and network switches to meet the overall demands so as not to overload certain resources.

The next challenge would be to virtualize cloud infrastructure! By decoupling infrastructure management from service management, cloud resources such as servers and networks can be provided with a pay-as-you-go model. Then, cloud service operators are liberated from hardware resource management, as was the case for system administrators by cloud computing. Moreover, it is not enough to shift responsibility to operators of the new lower infrastructure layer. We need a new self-managing infrastructure layer which does not require human operators.

We also consider two recent trends in cloud computing. One trend is edge computing [4] and micro datacenters [6]. Small-scale datacenters with a few racks or even a couple of servers can be placed at locations closer to the users, to reduce latencies or to store sensitive data on premise, and make them act as part of the cloud. A possible future direction is distributed cloud computing that utilizes diverse and geographically scattered computing resources. It will require a new management mechanism to efficiently utilize such resources. On the other hand, computing resources will be abundant in the environment so that it will be no longer necessary to micromanage computing resources.

Another trend is microservices [14] and serverless computing [17] in which a cloud service is composed of a collection of loosely-coupled lightweight services. Each microservice is ephemeral and short-lived, and can be executed in a stateless container, which enables flexible and efficient use of underlying cloud resources. The concept has something in common with early packet switching so that it might open up new possibilities to apply packet switching techniques to handling microservice jobs.

In addition, we should take energy saving into consideration as it is essential for future clouds [11, 12].

1.2 Cloud Morphing Vision

Cloud Morphing is our vision for cloud virtualization in the future. By dynamically allocating microservices over distributed heterogeneous resources, a cloud service instance emerges at the best location and, as the usage pattern changes, the service instance also transforms the locations of the resources and their connections.

Microservice jobs are assigned to reduce the execution cost that consists of computing cost, communication cost with the user, access cost to database, and other factors. For examples, an interactive task will follow the user when the user moves, while a data-intensive task will stay close to the data regardless of the user location. Edge computing is automatically formed by allocating resources close to the users. Moreover, services are inherently fault-tolerant and resilient against outages or disasters since faulty resources are automatically evicted from the resource pool.

From the operational perspective, the physical resource management becomes simpler with a larger resource pool. Physical nodes can be easily attached to or detached from the resource pool. When there is a consistent hotspot, it can be alleviated or solved by placing new resources close to the hotspot and then attaching them to the resource pool at a convenient time.

Diverse resources would be owned and managed by different parties. It requires loose management of resources, as small parties cannot afford dedicated skilled operators. The utilization of each resource needs to be easily manipulated, without affecting the stability of the system.

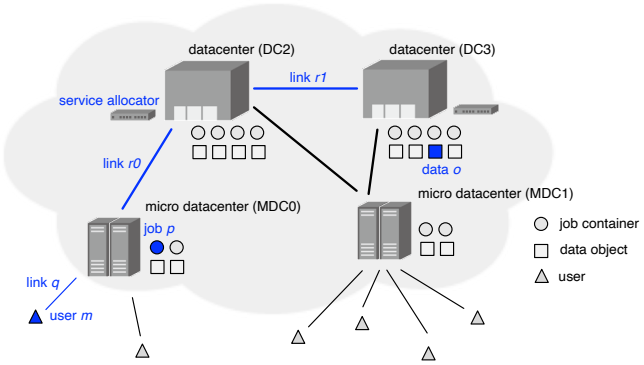

Figure 1: Simple system model for job assignment

To realize such systems, it requires various technical advances across many fields. Among other things, new autonomous resource management model is needed for distributed diverse resources, which is the topic of this paper.

1.3 Resource Management

Resource allocation in distributed clouds is a non-trivial optimization problem, and it is even harder when distributed management is assumed.

To this end, we employ dynamic pricing for decentralized resource allocation which works as backpressure against congestion. By design, serious congestion never happens in the system as long as some resources remain available in the resource pool. It also reserves a sufficient margin essential for the performance of statistically multiplexing services.

We use pseudo cost for manipulating resource allocation. Here, pseudo cost is not an actual monetary charge, but it is used to control resource utilization.

There exists a large body of literature formulating resource allocation as optimization problems. In this paper, we use terms and expressions borrowed from optimization theory. Our goal is, however, not to pursue theoretical optimum allocations, but to present a practical feedback control model for distributed resource management.

2 RESOURCE MANAGEMENT MODEL

Our resource management model is explained using a simple system configuration with 2 datacenters (DCs) and 2 micro-datacenters (MDCs) shown in Figure 1. When a user requests a service to a nearby service allocation server, the service server instantiates the requested service into a series of microservice jobs. The server obtains the required resource information for each job, identifies the locations of the user and the required data object, asks nearby resource agents for available resources and their current costs, and assigns the job to the node that minimizes the total cost for the job. The area for cost inquiry could be within proximity along the path between the user and the data object.

The load of a resource can be measured in a number of ways. Most resources have some form of load report functions built-in. When the load of a resource fluctuates too rapidly, a smoothing filter (e.g., exponentially weighted moving average) can be used to stabilize the behavior. A load value does not need to be precise, and a rough approximation is enough for loose resource management. Still, if each allocation is not small enough both in size and in duration, the load may not converge as expected. Therefore, microservice is the enabler for this approach.

2.1 Micro-job Assignment

In this paper, we use a simple micro-job model that defines the required resources for a micro-job as $J(p, q, r, s)$ where p is the number of micro containers for computation, q is frontend communication with the user, r is backend communication with data objects (e.g., database), and s is the number of time slots. The communication costs q and r are also a function of distance so that an interactive job with $q \gg r$ will be placed close to the user and a data-intensive jobs with $q \ll r$ will be placed close to the data. For simplicity, we do not distinguish directions of communications for q and r, and assume only one user and one data object per micro-job in this paper.

A micro-job is specified by a service provider, optionally associated with weights, e.g., a service provider may raise the weight for the frontend communication to place the job close to the user. In this manner, cloud service providers can specify which type of resources have priority for a specific service without overclaiming required resources.

To instantiate a micro-job requested from a user, the service server finds the best node to allocate the required resource for J: p, q and r for duration s.

The pseudo cost E to host micro-job j for a unit time at node i for user m and data object o is:

$$E(j, i) = H(j, i) + G(j, i, m, o)$$

here, $H(j, i)$ is the computing cost to run j at i, and $G(j, i, m, o)$ is the communication cost to run j at i between m and o.

$$H(j, i) = p \cdot f(\rho_i)$$
$$G(j, i, m, o) = q \cdot \sum_{l \in path(m,i)} f(\rho_l) + r \cdot \sum_{l \in path(i,o)} f(\rho_l)$$

$f(\rho)$ is the cost function of a resource load, and $path(m, i)$ is a set of links from m to i (e.g., the shortest path weighted by cost).

To assign micro-job j, the server simply finds the node that minimizes the cost:

$$argmin_i \, E(j, i)$$

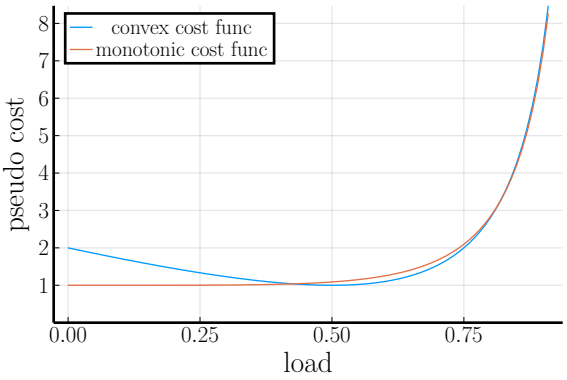

Figure 2: Standard cost functions

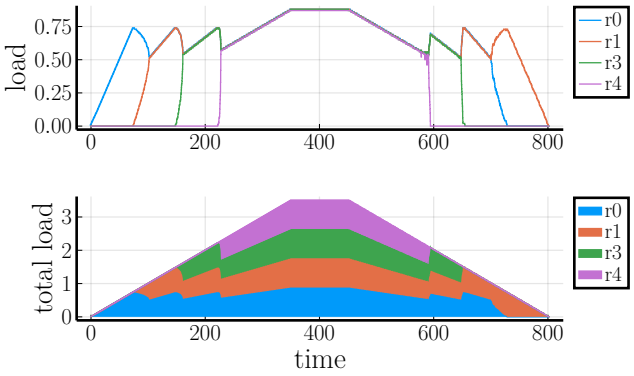

Figure 3: Load distribution among 4 equivalent cost nodes: the load of each resource (top) and the total load (bottom)

2.2 Pseudo Cost Functions

Our model employs a parametric representation of pseudo cost, as a function of load, and uses it for load control and also as backpressure against congestion. Micro-jobs are naturally gravitated to the most cost-efficient location.

The proposed model is based on congestion pricing in which the cost of a resource dynamically changes according to the load of the resource. It works as a barrier function for optimization; the capacity constraint is enforced by a penalizing cost when approaching the full capacity.

Another key idea is *idle-resource pooling* that tries to put resources into an idle state when possible for energy saving. We proposed a convex cost function that enables idle-resource pooling as part of the congestion pricing mechanism.

A pseudo cost function in our model maps the load of a resource $\rho \in [0, 1]$ to the corresponding cost. The capacity limit is enforced by the cost function that rapidly grows as the load approaches 1.0, which is known as a barrier function in optimization theory.

We use two types of pseudo cost functions: one is the monotonic cost function and the other is the convex cost function. The monotonic cost function is a simple barrier function that monotonically grows with load, up to infinite as $\rho \to 1$. The convex cost function is also a barrier function but also for idle-resource pooling. In this paper, the monotonic form is used for network links as energy-saving-by-idling is not common for network links.

The standard forms that have the minimum cost of 1.0 are shown in Figure 2. We will show how to manipulate the cost functions in Sec. 2.4.

The standard convex cost function is defined as:

$$f(\rho) = \frac{(2\rho - 1)^2}{1 - \rho} + 1$$

This function has the properties: $min\ f(\rho) = f(.5) = 1$, and $f(0) = f(.75) = 2$. The cost grows rapidly when $\rho > .75$. The system automatically tries to keep $\rho \leq .75$, aiming at $\rho = .5$.

The standard monotonic cost function is defined as:

$$f(\rho) = \frac{\rho^{4.5}}{1 - \rho} + 1$$

to roughly match the standard convex cost function in $[.5, .75]$, the *working load range* explained in the next subsection.

Note that the standard forms are defined just for convenience, and other functions with a similar shape in $[0, 1]$ also work for our purposes.

2.3 Idle-Resource Pooling in Action

The behavior of idle-resource pooling by the convex cost function is illustrated by the following examples in Figure 3 and Figure 4.

First, assume a pool of 4 equivalent resources with the standard convex cost function. Also, assume that micro-jobs are continuously assigned to the system; each micro-job is much smaller than the capacity of a resource.

The initial system load $\sum \rho$ is 0, and gradually increased up to 3.5 until time 350. After time 450, the system load is gradually decreased back to 0 until time 800. Here, load 1.0 is the capacity of a single resource. Initially, all resources in the pool are idle, and their costs are all $f(0) = 2$. For the first job, one resource r_0 is randomly selected for allocation, and its cost becomes lower: $f(0+) < 2$. As a result, subsequent jobs are assigned to r_0, with lowering cost towards $\rho = .5$ and then rising again until $\rho = .75$ where $f(.75) = 2 = f(0)$. At this point, another resource r_1 is selected for allocation. r_1 is preferred over r_0 as its cost becomes lower with new allocations so that both loads move towards $\rho_0 = \rho_1 = .5$, where both are balanced. Both loads rise again until $\rho_{r0} =$

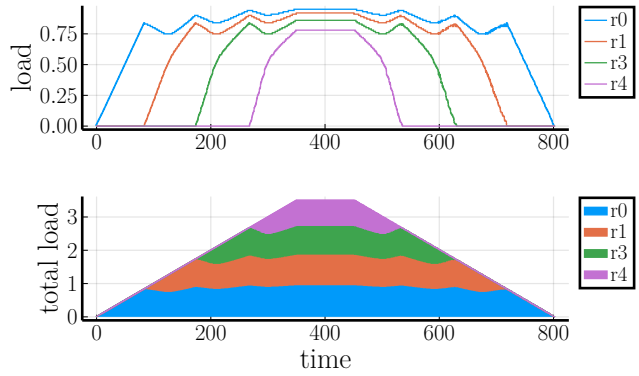

Figure 4: Load distribution: 4 proportional cost nodes

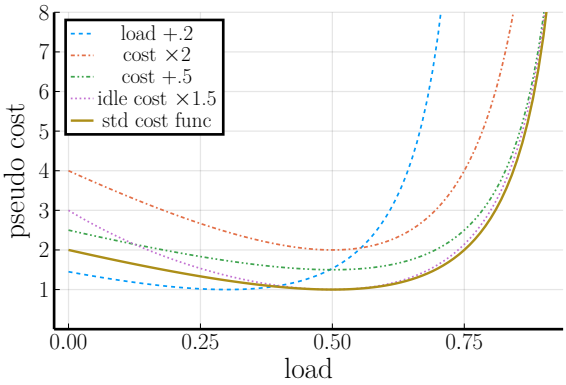

Figure 5: Manipulating convex cost functions

$\rho_{r1} = .75$, where the third resource r_2 kicks in. It repeats for r_3, but no more idle resource is available when $\sum \rho$ reaches 3.0 so that the loads grow beyond .75 up to $\sum \rho = 3.5$ with $\rho = .875$ for each.

When the system load decreases, the process is reversed. After reaching $\rho = .5$ for all, one resource with the lowest load $\rho < .5$ becomes more expensive than the others. This one is less preferred for subsequent assignments, and quickly loses the load until it becomes idle again, while the other 3 keep ρ in $[.5, .75]$. It repeats for the remaining ones.

It is easy to see how the number of active resources changes when there are more resources. When the number of active resources is increasing, the load of each active resource stays at around $\rho = .75$. On the other hand, when the number of active resources is decreasing, the load of each active resource stays at around $\rho = .5$. In short, when all resources are equal, the system tries to maintain the load of active resources in the *working load range* $[.5, .75]$, while keeping idle resources as much as possible.

When resources are not equal, the behavior becomes more complex, but the underlying mechanisms are the same. Let's take a look at a case of 4 resources in Figure 4 with the cost ratio $1 : 2 : 4 : 8$, that is $8f_{r0} = 4f_{r1} = 2f_{r2} = f_{r3}$. Here, we focus on the interaction between r_0 and r_1 since the other interactions are similar. To activate r_1, the load of r_0 goes up to .84 to satisfy: $f_{r1}(0) = 4 = f_{r0}(.84)$. When r_1 is moving towards idle after time 700, the load of r_0 is .75 to satisfy: $f_{r1}(.5) = 2 = f_{r0}(.75)$

For unequal resources in general, the required load to trigger a new allocation is higher than $\rho = .75$ for the already active ones to match the cost $f(0)$ for the new one, but the load will not go much further as the slope of the cost function is steep. Similarly, when the most expensive one among active resources becomes idle, the load of the remaining ones stay at the matching cost $f(.5)$ for the deactivating one.

2.4 Manipulating the Cost Function

The utilization of a resource can be manipulated by modifying the cost function of the resource as shown in Figure 5.

One can **lower or raise the load level** of a resource by shifting the load in the cost function and adjusting the target load, $f'(\rho) = f(\rho + \Delta)$.

To **change the activation order** in the idle-resource pooling, one can raise or lower the cost, e.g., by making the cost n times more expensive, $f'(\rho) = nf(\rho)$. For minor adjustment, one can use an additive form, $f'(\rho) = f(\rho) + \Delta$.

To make **idle-resource pooling more aggressive**, one can raise the cost at $\rho = 0$: e.g., to raise the cost at $\rho = 0$ by a factor of $(n+1)/2$, $f'(\rho) = n(2\rho - 1)^2/(1 - \rho^n) + 1$.

A **premium service** can be realized by shifting the load in the same way as lowering the load level but for specific users or jobs (not for a specific resource) so as to have premium jobs always being assigned to less loaded resources. Similarly, an **economy service** that allows to be assigned to more loaded resources can be made by a negative shift. It could be appealing as a business model to enable multiple classes using a single resource pool with a single shift parameter.

Other than manipulating the cost function, cloud service providers can adjust the required resources and their weights for a job. Also, it is effective to place data objects close to the users, and both service providers and their users should have some control over where to store the data. Thus, the system allows stakeholders to loosely control the resource utilization.

3 SIMULATION RESULTS

We have developed a simple simulation tool to evaluate the model that is publicly available from (URL removed for blind-review). We omit the details of the simulation settings in this paper but all the settings are described in the simulation tool.

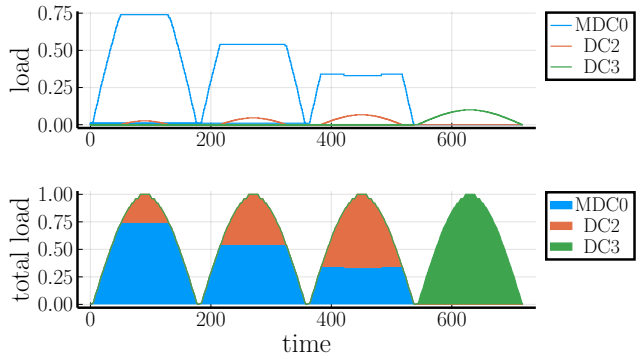

Figure 6: Cost manipulations: shifting the load +.2, +.4 and raising the weight for data access

3.1 Basic Behaviors

The effects of manipulating the cost function and the resource weight for a job is illustrated in Figure 6 using the topology in Figure 1. The capacities of MDC and DC are set to 100 and 1,000. Note that the capacity of DCs is 10 times larger than that of MDCs so that DC's load in the top plot looks much smaller for the same volume of jobs. In the normalized total load in the bottom plot, the volume is normalized to the capacity C of MDC to show the total volume of jobs: $\sum \tilde{\rho}$, $\tilde{\rho} = \rho \cdot C/C_{MDC}$.

Here, a series of interactive jobs of the same type are generated in 4 waves between a user at MDC0 and an object at DC3. In the first wave, most of the jobs are assigned to the node closest to the user (MDC0) but some overflowing jobs are offloaded to DC2, the upstream of MDC0. The peak load of MDC0 is .74 in the first wave. In the second and the third waves, the cost function of MDC0 is manipulated to reduce the load by shifting the load by .2 and .4 respectively. As a result, the peak load of MDC0 is reduced from .74 to .54 in the second wave, and then, to .33 in the third wave. In the fourth wave, the weight for the backend communication is increased, and all jobs are assigned to the node that has the object (DC3).

3.2 Mixed Behaviors

A more complex scenario is shown in Figure 7. Again, the topology in Figure 1 is used, and random fluctuation is added to the interval and duration of jobs. A series of jobs are generated between user0 at MDC0 and an object at DC3, and between user1 at MDC1 and an object at DC2. Both have the ratio of 1 : 2 for interactive vs. data-intensive jobs. To observe offloading behaviors, user1's jobs are increased by a factor of 2 in the third wave (time 360-540), and by a factor of 10 in the fourth wave (time 540-720). Before time 360, interactive jobs are assigned closer to the users, and data intensive jobs

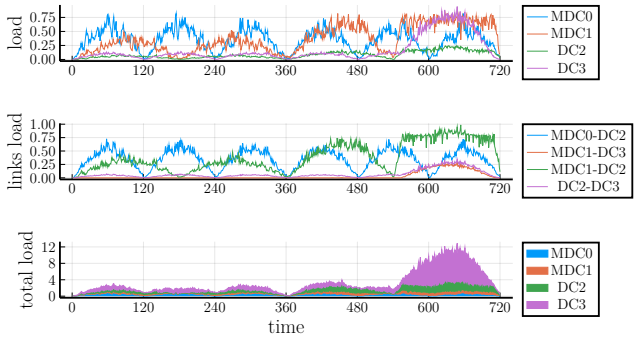

Figure 7: Mixed load with 2 DCs and 2 MDCs

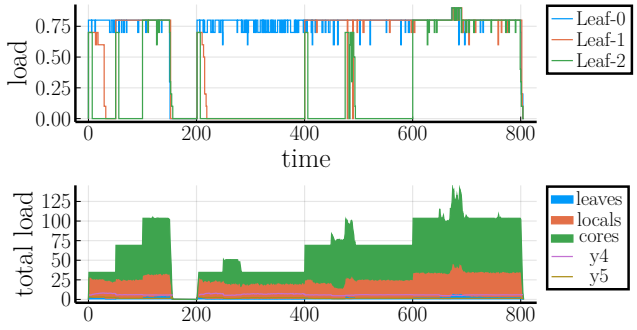

Figure 8: Mixed load with 18 DCs

are assigned closer to the data. In the third wave, MDC1 reaches the upper limit and overflowing jobs are assigned to DC2. In the fourth wave, the link MDC1-DC2 reaches the upper limit so that overflowing jobs are assigned to DC3. (To absorb the surge in the fourth wave in this scenario, enough capacity is provided to the link MDC1-DC3.) This scenario illustrates the responsive behavior of the system, and how jobs for MDCs can be offloaded to upstream DCs.

The last scenario in Figure 8 shows a more complex case with 18 DCs: 2 core-DCs (capacity 5,000), 8 local-DCs (capacity 500) and 8 leaf-DCs (capacity 100) connected by a FAT-tree. Each leaf-DC is connected to one local-DC, and each local-DCs is connected to 2 other local-DCs; 4 of the local-DCs are also connected to 2 core-DCs. The bottom plot shows the normalized total load $\sum \tilde{\rho}$ divided into the 3 DC classes. Jobs are generated from 3 leaf-DCs at 3 levels in volume with additional surges, and data objects are randomly populated at all leaf-DCs and local-DCs. The bottom plot shows how excess jobs are offloaded to upstream DCs, and the top plot shows the loads of the 3 job-generating leaf-DCs. Although the leaf-DCs handle only a small fraction of the total jobs, the loads of the leaf-DCs stay at around .75 while

they are active, except a short transient overshoot during the surge from time 680.

Overall, the simulation results illustrate the basic behaviors of the idle-resource pooling, quick responses to changing load by offloading excess jobs to upstream DCs, and converging to stable states once the load is settled while maintaining the load for each active resource at around the target load.

4 RELATED WORK

Congestion pricing and smart data pricing for networking have been extensively studied [16]. Early work includes congestion pricing by Murphy *et al.* [13], and auction-based resource allocation by MacKie-Mason *et al.* [10], both in 1994. Kelly *et al.* [8] framed congestion pricing in an optimization framework for fair allocation, which inspired a large number of the following work [2, 5, 7, 16]. Congestion pricing are also applied to cloud computing [9, 18, 19, 22].

Resource allocation for edge computing and micro data centers also has a rich collection of work; most approaches are based on some form of optimization and many employ auction models. The topics are ranging from VM auctions in IaaS [24, 25], load balancing within a data center [3, 15], to distributed resource management for microservices [20].

Xu *et al.* [23] proposed the edge computing infrastructure layer similar to ours, separating the cloud infrastructure service from the cloud service for micro datacenters, but an auction model is proposed for resource allocation.

Exponential cost growth as a function of load is well known in packet switching networks (e.g., M/M/1 queue and CSMA/CD Ethernet). The idea to use such cost functions for resource management was in [13] where Murphy *et al.* used a cost function for distributed bandwidth allocation in ATM networks in 1994. Their cost function is a barrier function for utility optimization and their simple cost minimizing allocation algorithm is also somewhat similar to our allocation model.

A system model similar to ours is found in [21] where Wagner *et al.* used congestion pricing for resilient job allocation in a distributed military cloud. They used a game-theoretic resource allocation method based on Nash Bargaining, and developed a Hadoop-based prototype system.

We were inspired by the concept of micro cloud services to apply packet switching techniques to distributed heterogeneous clouds, and have revisited congestion pricing. To the best of our knowledge, our work is unique in using a convex cost function for idle-resource pooling.

5 SUMMARY AND FUTURE WORK

In this paper, we have presented the cloud morphing vision for future distributed cloud services, proposed a resource allocation model based on cost functions, and presented how the idle-resource pooling works. We are planning to refine the proposed model and develop a working prototype system.

For the model refinement, the current model is simplistic so that we will add bidirectional communication costs, multiple users per job, interactions among jobs, and other features. We did not consider dependency among microservice jobs, but it would be necessary to investigate the impact of the interplay of microservice jobs [20].

For the prototype development, we need realistic future microservice workload models. Also, it is necessary to develop mechanisms and protocols for discovering available resources [1] and exchanging cost information. A path selection mechanism is also needed when assigning a job, probably using a source routing mechanism.

There exists a rich area for further research: one topic is **auto-tuning of the cost functions**. It may not be straightforward to obtain expected results by tuning the convex cost function due to the interactions between load-balancing and idle-resource pooling. It would be interesting to apply reinforcement learning to cost tuning.

Another topic is **hierarchical configurations**. A hierarchical system model would be preferred for scalability and for management purposes. For example, a distributed datacenter model can consist of the inter-DC layer with DC-level resources and the intra-DC layer with rack-level resources.

The proposed system was originally designed mainly for a single administrative domain. The hierarchical model, however, naturally extends to an **inter-cloud model** in which different cloud systems are federated. The mechanism of the idle-resource pooling allows utilizing external clouds only when needed.

Further, it would be possible to **crowdsource the resource supply** at the edge. Then, we need a monetary charging, authentication and authorization mechanisms to add externally provided resources to the resource pool.

The location of data is critical for the pseudo cost so that **data placement** would play an important role for cloud morphing. A possible approach is to migrate data at a coarser timescale, based on usage history. Another possibility is to design a new distributed storage system suitable for the cloud morphing model in which (cached) data can be easily moved.

The idle-resource pooling is not used for network links in this paper, but it is possible to use it for dynamically making **Layer 2 paths** (e.g., lightpath switching over WDM networks).

We believe that future distributed heterogeneous clouds need a new paradigm for resource management, and hope this work will stimulate other research in the field.

REFERENCES

[1] Jeannie Albrecht, David Oppenheimer, Amin Vahdat, and David A. Patterson. 2008. Design and Implementation Trade-Offs for Wide-Area

Resource Discovery. *ACM Trans. Internet Technol.* 8, 4, Article 18 (oct 2008), 44 pages. https://doi.org/10.1145/1391949.1391952

[2] Bob Briscoe, Vasilios Darlagiannis, Oliver Heckman, Huw Oliver, Vasilios Siris, David Songhurst, and Burkhard Stiller. 2003. A Market Managed Multi-Service Internet (M3I). *Comput. Commun.* 26, 4 (mar 2003), 404–414. https://doi.org/10.1016/S0140-3664(02)00158-5

[3] Liuhua Chen, Haiying Shen, and Karan Sapra. 2014. Distributed Autonomous Virtual Resource Management in Datacenters Using Finite-Markov Decision Process. In *Proceedings of the ACM Symposium on Cloud Computing* (Seattle, WA, USA) *(SOCC '14)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/2670979.2671003

[4] Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. 2015. Edge-Centric Computing: Vision and Challenges. *SIGCOMM Comput. Commun. Rev.* 45, 5 (sep 2015), 37–42. https://doi.org/10.1145/2831347.2831354

[5] Richard J Gibbens and Frank P Kelly. 1999. Resource pricing and the evolution of congestion control. *Automatica* 35, 12 (1999), 1969–1985.

[6] Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel. 2009. The Cost of a Cloud: Research Problems in Data Center Networks. *SIGCOMM Comput. Commun. Rev.* 39, 1 (dec 2009), 68–73. https://doi.org/10.1145/1496091.1496103

[7] T. Henderson, J. Crowcroft, and S. Bhatti. 2001. Congestion pricing. Paying your way in communication networks. *IEEE Internet Computing* 5, 5 (2001), 85–89. https://doi.org/10.1109/4236.957899

[8] F.P. Kelly, A.K. Maulloo, and D. Tan. 1998. Rate Control for Communication Networks:Shadow Prices, Proportional Fairness and Stability. *Journal of the Operational Research Society* 49 (02 1998). https://doi.org/10.1057/palgrave.jors.2600523

[9] Cinar Kilcioglu and Costis Maglaras. 2015. Revenue Maximization for Cloud Computing Services. *SIGMETRICS Perform. Eval. Rev.* 43, 3 (nov 2015), 76. https://doi.org/10.1145/2847220.2847245

[10] Jeffrey K. MacKie-Mason and Hal R. Varian. 1994. *Pricing the Internet.* Computational Economics 9401002. University Library of Munich, Germany. https://ideas.repec.org/p/wpa/wuwpco/9401002.html

[11] Eric Masanet, Arman Shehabi, Nuoa Lei, Sarah Smith, and Jonathan Koomey. 2020. Recalibrating global data center energy-use estimates. *Science* 367, 6481 (2020), 984–986.

[12] Toni Mastelic, Ariel Oleksiak, Holger Claussen, Ivona Brandic, Jean-Marc Pierson, and Athanasios V. Vasilakos. 2014. Cloud Computing: Survey on Energy Efficiency. *ACM Comput. Surv.* 47, 2, Article 33 (dec 2014), 36 pages. https://doi.org/10.1145/2656204

[13] John Murphy, Liam Murphy, and Edward C. Posner. 1994. Distributed Pricing For Embedded ATM Networks. In *The Fundamental Role of Teletraffic in the Evolution of Telecommunications Networks*, JACQUES LABETOULLE and JAMES W. ROBERTS (Eds.). Teletraffic Science and Engineering, Vol. 1. Elsevier, 1053–1063. https://doi.org/10.1016/B978-0-444-82031-0.50108-6

[14] Irakli Nadareishvili, Ronnie Mitra, Matt McLarty, and Mike Amundsen. 2016. *Microservice architecture: aligning principles, practices, and culture.* " O'Reilly Media, Inc.".

[15] Negar Rikhtegar, Manijeh Keshtgari, Omid Bushehrian, and Guy Pujolle. 2021. BiTE: a dynamic bi-level traffic engineering model for load balancing and energy efficiency in data center networks. *Appl. Intell.* 51 (2021), 4623–4648.

[16] Soumya Sen, Carlee Joe-Wong, Sangtae Ha, and Mung Chiang. 2013. A Survey of Smart Data Pricing: Past Proposals, Current Plans, and Future Trends. *ACM Comput. Surv.* 46, 2, Article 15 (nov 2013), 37 pages. https://doi.org/10.1145/2543581.2543582

[17] Hossein Shafiei, Ahmad Khonsari, and Payam Mousavi. 2022. Serverless Computing: A Survey of Opportunities, Challenges, and Applications. *ACM Comput. Surv.* (jan 2022). https://doi.org/10.1145/3510611 Just Accepted.

[18] Jiayi Song and Roch Guerin. 2017. Pricing and bidding strategies for cloud computing spot instances. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 647–653. https://doi.org/10.1109/INFCOMW.2017.8116453

[19] Shaoyi Song, Tingjie Lv, and Xia Chen. 2014. A cooperative game method for load balancing in cloud based on cost-efficiency. In *2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, 311–314. https://doi.org/10.1109/ICUFN.2014.6876803

[20] Lalith Suresh, Peter Bodik, Ishai Menache, Marco Canini, and Florin Ciucu. 2017. Distributed Resource Management across Process Boundaries. In *Proceedings of the 2017 Symposium on Cloud Computing* (Santa Clara, California) *(SoCC '17)*. Association for Computing Machinery, New York, NY, USA, 611–623. https://doi.org/10.1145/3127479.3132020

[21] Stuart Wagner, Eric Van Den Berg, Jim Giacopelli, Andrei Ghetie, Jim Burns, Miriam Tauil, Soumya Sen, Michael Wang, Mung Chiang, Tian Lan, Robert Laddaga, Paul Robertson, and Prakash Manghwani. 2012. Autonomous, Collaborative Control for Resilient Cyber Defense (ACCORD). In *2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems Workshops*. IEEE, 39–46. https://doi.org/10.1109/SASOW.2012.16

[22] Hongyi Wang, Qingfeng Jing, Rishan Chen, Bingsheng He, Zhengping Qian, and Lidong Zhou. 2010. Distributed Systems Meet Economics: Pricing in the Cloud. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing* (Boston, MA) *(HotCloud'10)*. USENIX Association, USA, 6.

[23] Jinlai Xu, Balaji Palanisamy, Heiko Ludwig, and Qingyang Wang. 2017. Zenith: Utility-Aware Resource Allocation for Edge Computing. In *2017 IEEE International Conference on Edge Computing (EDGE)*. IEEE, 47–54. https://doi.org/10.1109/IEEE.EDGE.2017.15

[24] Sharrukh Zaman and Daniel Grosu. 2013. A Combinatorial Auction-Based Mechanism for Dynamic VM Provisioning and Allocation in Clouds. *IEEE Transactions on Cloud Computing* 1, 2 (2013), 129–141. https://doi.org/10.1109/TCC.2013.9

[25] Xiaoxi Zhang, Zhiyi Huang, Chuan Wu, Zongpeng Li, and Francis C. M. Lau. 2017. Online Auctions in IaaS Clouds: Welfare and Profit Maximization With Server Costs. *IEEE/ACM Transactions on Networking* 25, 2 (2017), 1034–1047. https://doi.org/10.1109/TNET.2016.2619743