Entropy4Cloud: Using Entropy-Based Complexity to Optimize Cloud Service Resource Management

Huankai Chen, *Student Member, IEEE*, Frank Z. Wang, *Senior Member, IEEE*, and Na Helian

***Abstract*—In cloud service resource management system, complexity limits the system's ability to better satisfy the application's quality of service requirements, e.g., cost budget, average response time, and reliability. Numerousness, diversity, variety, uncertainty, etc., are some of the complexity factors that lead to the variation between expected plan and actual running performance of cloud applications. In this paper, after defining the complexity clearly, we identify the origin of complexity in cloud service resource management system through the study of "Local Activity Principle." In order to manage complexity, an entropy-based methodology is presented to use, which covers identifying, measuring, analyzing, and controlling (avoid and reduce) of complexity. Finally, we implement such idea in a popular cloud engine, Apache Spark, for running analysis as a service. Experiments demonstrate that the new entropy-based resource management approach can significantly improve the performance of spark applications. Compare with the fair scheduler in Apache Spark, our proposed entropy scheduler is able to reduce overall cost by 23%, improve the average service response time by 15–20%, and minimized the standard deviation of service response time by 30–45%.**

***Index Terms*—Entropy theory, complexity, cloud services, resource management.**

I. Introduction

RESOURCE Management is an NP-complete problem, the complexity of which increase substantially when applications are deployed in the cloud. The complexity of cloud service resource management may originate from many factors: the scale of resource size; the heterogeneity of resource types and their interdependencies; as well as the variability, dynamicity and unpredictability of resource run-time performance.

Complexity has many negative effects on satisfying the Quality of Service (QoS) requirements of Cloud applications such as cost, performance, availability and reliability. If an application can not guarantee its QoS, it will be hard to populate its service. However, the vast majority of research efforts in Cloud service resource management implicitly assume the Cloud to be simplify and the Cloud resource's performance is determined and predictable. The improper assumption may significantly affect the QoS of the Cloud application and cause resource management strategy to be less robust.

In spite of extensive research of complexity issues in different fields ranging from computational biology to decision making in economies, a study of complexity for Cloud service resource management system is limited. In this paper, we address these complexity problems in Cloud Service Resource Management System by introducing *Entropy Theory*. The main contributions of this paper are as follows:

1) Complexity in the *Cloud Service Resource Management System* is clearly defined, which origin is identified through the study of "Local Activity Principle".
2) *Entropy Theory* based methodology for resource management in cloud service is proposed to use which cover identifying, measuring, analysing and controlling (reduce and avoid) of complexity.
3) After figure out the root cause of complexity by using "Local Activity Principle", Resource Entropy Based Local Activity Ranking is proposed to solves the resource management problem by controlling the *Local Resource Complexity*. Finally, we implement such idea named "Entropy Scheduler" in a popular real-world cloud analysis engine, Apache Spark. Experiments demonstrate that the new "Entropy Scheduler" outperform the default "Fair Scheduler" for better quality of service satisfaction.

In this paper, we discuss the complexity measurement base on Entropy Theory, which can be simply applied in the cloud service resource management system. Section II define the related concept of Complexity in the cloud. Then, Section III introduce the basic Entropy Theory and describes how Entropy is used to control the complexity in cloud resource management system. Finally, Section IV evaluates the real world cloud applications based on the proposed Entropy based methodology and discuss the experimental results. Section V describes related work, and Section VI presents our conclusions and future work.

II. Complexity in the Cloud

For now, the concept of complexity with respect to the cloud has not been precisely delineated yet. Despite the fact that the concept of complexity is somewhat ambiguous and varies from author to author, there are still several typical properties being shared by numerous complex systems.

1) *Complex systems are made up of several non-linear components*
   A cloud resource management system's resources serve as the cloud's basic components. These resources are

Manuscript received February 28, 2017; revised May 17, 2017 and June 16, 2017; accepted June 27, 2017. Date of current version January 19, 2018. *(Corresponding author: Huankai Chen.)*

H. Chen is with the Data Science Group, School of Computing, University of Kent, Canterbury CT2 7NF, U.K., on leave from Tsinghua University, Beijing 100084, China (e-mail: HC269@kent.ac.uk).

F. Z. Wang is with the University of Kent, Canterbury CT2 7NF, U.K. (e-mail: F.Z.Wang@kent.ac.uk).

N. Helian is with University of Hertfordshire, Hatfield AL10 9EU, U.K. (e-mail: n.helian@herts.ac.uk).

Digital Object Identifier 10.1109/TETCI.2017.2755691

non-linear. During run-time, the performance of the resource is highly dynamic and is influenced by the running jobs. Non-linearity is a condition that is needed for chaos. Furthermore, almost every system having a phase space with three or more dimensions can be considered chaotic in a certain part of that phase space [1].

2) *A complex system's components are interdependent*
The cloud's resources indirectly interact with each other via the resource management system. The state of the resources depends on other resources and is affected by the state of the other resources as well.
3) *A complex system has a structure that spans several scales*
For example, let us examine a typical cloud resource management system:
   - Scale 1: resource management; applications; resources;
   - Scale 2: resource allocation, job scheduling; jobs, subtasks; hardware, software ...
   - Scale 3: constrains, objects; parameters, functions, variables, requirements; CPU, operating system, memory, storage ...
   - More scales : ...

   Every scale has a structure. This complex system's essential and virtually new aspect allows the system to be capable of handling emerging behaviour.

1) *A complex system can handle emerging behaviour*
Emergence takes place when the focus of attention is shifted from one scale to another coarser scale above it. Observed at a specific scale, a certain behaviour is considered emergent if one cannot understand it after studying it separately and one by one. Each of this scale's components may also be a complex system that comprises finer scale. Therefore, the emerging behaviour is a novel phenomenon that is special to the scale being studied. Moreover, it is a result of the global interaction between that scale's components [1]. For instance, a computer has the ability to run a program, which is the highest scale's emerging behaviour. If the study is only focused on lower scale components like the transistor, wire, or power, one will never get an understanding of how the computer runs the program.
2) *Complexity involves an interaction between chaos and order*
It has been said that many complex systems do not always display chaos at all times. In other words, they display chaos for some of the control parameter's values, but also display order for others. Furthermore, there is the edge of chaos, i.e. the control's precise value when the system's state switches between chaos and order.
3) *Complexity involves an interaction between competition and cooperation*
Within the cloud, resources work together to complete the job. However, they also compete for the job's sub-tasks according to their states.

Seen from a global view, a cloud service resource management system is made up of numerous resources which collaborate either directly or indirectly in order to meet the application requirements. These resources and the interrelationships among them are important for the complexity that takes place in that system. Seen from a local view, the actual resource itself displays various degrees of complexity as well. These degrees of complexity come from either internal sources (memory, disk, CPU, etc.) and/or external sources (jobs running on it).

A. Characteristic of Complexity

The complexity found in cloud resource management systems has some key characteristics. It is important to understand how these characteristics affect the occurrence of complexity, either from the local resources it manages or the global system itself. However, these characteristics can act on one another or on each other. Therefore, explanations of these characteristics do not only represent the actual characteristic itself. Instead, it also emphasises the interaction and relationship among themselves.

1) *Numerousness* refers to the number of cloud resources that have to be managed by the system. A large number and a high level of the resources contribute to the system's increased complexity. Changes in the number of resources that are managed by the system under any consideration directly relate to any changes in the level of complexity.
2) *Diversity* is related to the cloud's homogeneity or heterogeneity. The resource's high/low diversity level can lead to heterogeneous/homogeneity and produces a high/low degree of complexity.
3) *Interdependency* refers to the intended or unintended relationship among cloud resource. This may lead to complexity within the management system. For instance, data required for a specific job can be partitioned or replicated onto multiple resources. These interdependent resources will not be able to perform the job without each other or without being influenced by each other. The increase of interdependence directly increases and affects complexity.
4) *Variability* refers to the changeability state, where an event leads to possible various outcomes in the local resource or global system. In terms of the global system, the resource state changes over time (e.g. performance, availability) and leads to a change in the capacity of the system. Seen from a local resource point of view, the change in its underlying components' states (e.g. memory consumption, CPU utilisation) leads to a change in its performance. Increasing the variability leads to a higher complexity level.
5) *Variety* is related to the state of being various. In making management decisions, the states of the system (e.g. under-provision/over-provision, number of resources, order/edge of chaos/chaos, under-loaded/over-loaded) and the state of resource (e.g. high/low CPU utilisation, number of free cores, high/low memory consumption ...) may have to be considered. This state variety represents the system or resource's dynamic behaviour. The more the states involved during decision making, the more the complexity that is introduced.
6) *Uncertainty* refers to all the difficulties experienced during the production of a clear picture of the resource or the system. This is caused by the lack of information. Uncertainty and complexity have a close relationship with one

another. More complexity occurs when there is more uncertainty within the cloud resource management system.

The complexity characteristics mentioned above can have close relationships with each other. In other words, one can influence the others or one can lead to the occurrence of the others. For instance, variability in the system may be created by a high level of variety or uncertainty can be caused by high density of diversity. However, the characteristics do not affect (more or less) the system with or without any interrelationships or interactions between them. Thus, generally, if these characteristics' level is reduced, the complexity will be reduced too.

III. Entropy-Based Complexity for Cloud Service Resource Management

Being able to manage the increasing complexity within the cloud service resource management system is needed to better satisfy the cloud applications' QoS requirements. In order to efficiently and effectively manage complexity, it is recommended that one need to identify, measure, analyse and control complexity first. Every one of the steps mentioned above is vital to complexity management. Measuring is the most important stage since it allows for the other stages to be performed effectively [2].

A. Identification

Identification is the first step in efficiently and effectively managing the complexity in cloud service resource management systems. This step is meant to determine the origin of complexity as well as the characteristics that are related to it.

1) Local Activity Principle: The local activity principle was originally from electronic circuits. However, it could be mathematically formulated in an axiomatic manner without having to mention any circuit models. For a spatially-extended dynamical system that is made up of more than one identical cell, changes in the state of the cell are dictated by a specific reaction-diffusion equation and the kinetic equations related to them. In other words, changes in the local cell state are influenced by some/all of the system's other cell states and by the cell's local diffusion in some cases. Since the role of the diffusion term in the reaction-diffusion equations is only a dissipative and stabilising one, the complex phenomenon observed in the system can only originate from the cell kinetic equations [3]. It can be proven rigorously that if there are no locally active cell kinetic equations, complexity cannot be exhibited by the reaction-diffusion equation. A cell that possesses a local-active kinetic equation can display complex dynamics like chaos or limit cycles, even if the cells are not couple to each other. Therefore, it is no surprise that coupling such cells could lead to an emerging pattern within the system. Thus, the cell that has a local-active kinetic equation is indeed the complexity's origin [4].

Definition of Local Activity: "A cell is considered locally active within the cell equilibrium point if, and only if, a continuous input time function exists in such a way that at some time point, no net energy is going out of the cell. The cell's initial energy is zero".

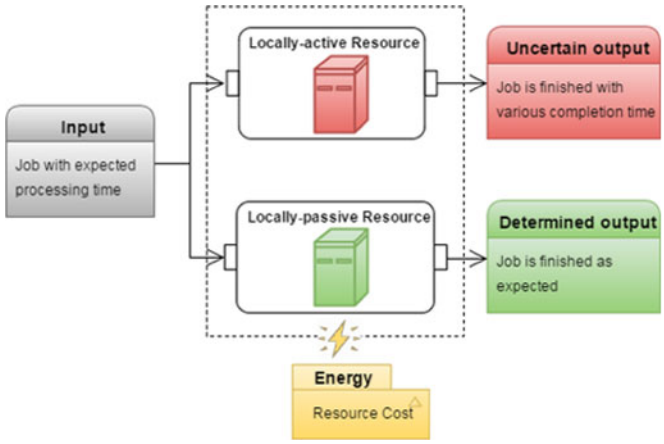


Fig. 1. Resource in the Cloud: locally-active vs. locally-passive.

Local Passivity Definition: "A cell is considered locally passive within the cell equilibrium point if, and only if, the cell stays at the initial state and has zero energy for all continuous input time functions".

Transistor is an typical example of a locally-active device. For the transistor, a low-power input signal can be turned into a high-power output signal. However, it is at the expense of an energy supply. Televisions, radios, or computers will not be able to function if they don't use locally-active devices like transistors. Moreover, any system that is made up of locally-active devices is considered locally active too.

The **Principle of Local Activity** is easily translatable into other non-electrical heterogeneous/homogeneous media. For cloud computing, an example of a locally-active device is the resource. As shown in Fig. 1, a small (estimated task's running time) input signal can be turned into a large (actual task's processing time) output signal. This conversion is at the expense of energy supply (resource cost). By definition, a resource that is not locally active is locally passive, in the sense that a resource having a fixed cost is guaranteed to offer a consistent performance during run-time. However, the resources of the cloud in the real world are rarely in the passive mode. In other words, they always display various degrees of local activity [5]. For instance, on average, a virtual resource has more less activity than a physical resource that has the same configuration. Moreover, for the same resource, the degree of activity varies during runtime.

2) Original of Complexity: Local Active Resource: Being the complexity's origin, the local activity resource directly affects the cloud resource management system's complexity level. For electronic circuits having homogeneous media, the system may fall in the "Edge of Chaos" when the locally active cells are within some parameter regions [6]. Furthermore, this will increase the probability of turning into completely chaotic. In the cloud environment, these complexity effects resulting from locally active resources will tend to take place more frequently. If the cloud service resource management system is within the chaotic state, its performance becomes harder to predict and it becomes degraded. Moreover, it fails to better meet the application's quality of service requirements. However, in most

past works, majority of the researchers do not consider the impacts that the resource's local activity has on the cloud resource management system. Instead, when developing a new management strategy, they assume that the resources are locally passive. Therefore, their research solution always fails to offer better QoS when they are run on real world cloud environments. The following are some of the complexity characteristics that are related to locally active resources:

1) *Heterogeneity:* Cloud systems can function like large virtual supercomputer. However, it could still have very disparate computational resources, ranging from laptops, clusters, desktops, supercomputers, and even smartphones that have limited computational power [7]. Current infrastructures for cloud technology are not very versatile yet. However, for the cloud system, heterogeneity is one of the most important features that have to be considered. With the advent and development of virtualisation technology, multiple VMs can be hosted on a single physical machine simultaneously. However, virtualisation can also lead to new challenges to the cloud's resource scheduling because of the fact that multiple VMs can share the physical machine's hardware resources (e.g. memory, hard disk, CPU, network) [8]. In a situation like this, accurately measuring the actual performance of the rented VMs can be difficult. For instance, in Amazon AWS, resource provisioning to vms has its basis in compute units instead of being based on fixed performance measures. Various host machines offer differing amounts of computing power per provisioned compute unit, leading to a heterogeneity in the VM performance [9]. This means that in the real world, the cloud should always be heterogeneous and could never be homogeneous.
2) *Dynamicity:* Another vital complexity factor that is inherent to the cloud environment is the dynamic changes within the resource performance during run-time [10]. Within the real world, this resource performance dynamicity can be a result of resource over- or under-provisioning, software/hardware failures, resource CPU overload, or even application misbehaviours. Furthermore, the cloud resource is influenced by the number of running jobs that are assigned to it as well. These jobs exhibit local activity and are considered the origin of complexity. Moreover, when it shares a common underlying hardware infrastructure with other virtual machines, the resource dynamicity will be brought up to a higher complex degree.
3) *Uncertainty:* A large majority of the past works conducted within the field of cloud resource management works on the assumption that there is complete knowledge about the cloud resource's run-time state. However, for cloud computing, the resource's computing capacity and ready time undergo considerable uncertainties during provisioning [11]. We contend that this uncertainty is the main inconvenience of the cloud because it brings about additional challenges that are involved when one has to predict the completion time of jobs. This is an important aspect of making scheduling decisions. In a cloud environment, resource states can change drastically. It is not possible to obtain an exact knowledge about the resource most of the time. It is difficult to accurately estimate the completion time of tasks, improve prediction using historical data, correct the prediction, have a prediction fall-back, etc. This inaccurate prediction execution results into an associated scheduling performance that is considerably uncertain.

B. Measurement

After the origin of complexity is identified, it is recommended that one has to provide a measurement which can dictate the behaviour of the locally active resource. Therefore, based on this studys definition of complexity, it is measured using entropy [12].

1) Entropy Theory: "Entropy is a vital statistical quantity that measures the degree of disorder and the amount of energy that is spend during the transformation from one state to another within the same system" [13]. Originally, the entropy concept was a thermodynamic construct. However, it has been applied to many other fields of research as well, including production planning, information theory, computer modelling and simulation, and resource management [14]–[18]. This measure will be used to quantify the reliability degree that is associated with the resource management system while using various strategies. We first introduce this Entropy measure in a general manner. Considering a dynamic system A in a finite and mutually exclusive state variable set $a = a_1, a_2, a_3, \ldots, a_n$ and having probabilities $p_1, p_2, p_3, \ldots, p_n$ respectively, entropy $H(A)$ can then be defined as:

$$H(A) = -\sum_{i=1}^{n} p_i * \log p_i \tag{1}$$

Given two dynamic systems that are mutually independent X and Y and that have n and m states, respectively, the probability $_{\text{ij}}$ of the concurrent occurrence of the states X_i and Y_j is $p_i q_j$. Here, p_i represents the probability of state i that takes place in system X, q_j represents the probability of state j that takes place in system Y, where $1in$ and $1jm$. Consequently, the sets of states $X_i Y_j$ are allowed to represent another finite system that can be designated by XY. Thus, we can say that:

$$H(XY) = H(X) + H(Y) \tag{2}$$

Where the corresponding entropies are $H(XY)$, $H(X)$ and $H(Y)$ are for systems XY, X and Y. Such an expression can be extended easily for an arbitrary number of finite systems that are mutually independent. On the other hand, given a system S that is made up of s mutually independent sub-systems $s_1, s_2, s_3, \ldots, s_k$, the entropy can be presented as:

$$H(S) = -\sum_{i=1}^{k} H(s) \tag{3}$$

Obtaining the average system entropy [18] can be done easily by:

$$\overline{H} = \frac{H(S)}{k} \tag{4}$$

This entropy measure's other properties, like those for dependent schemes, can also be found, for instance, in Khinchins paper [19]. For this study, only mutually independent systems will be considered.

C. Analysis

After measurements are done, the results obtained from the complexity measures will then have to be analysed. Analysing the value's complexity is related to the measurement goals. Analysing measurements can be done from many perspectives. For instance, one can implement a complexity measure for:

1) Analysing the resource's local activity level and making comparisons, among others.
2) Analysing the global system to determine whether it is in a state of order or chaos.

1) Degree of Local Activity: The local activity principle is the reason for breaking symmetry in homogeneous media. This serves as a rigorous but effective tool in identifying the resource's states. An increment in the local resource's activity will result into an increment of the complexity in the global system. This means that there is a higher chance for the system to fall into chaos.

Thus, we are introducing entropy as a quantitative measurement that can be used to make comparisons for the degree of local activity among the cloud resources. The goal of measuring local activity is to be able to produce a numerical scale that can be used to compare the activity degree on various cloud resources. In a practical sense, it is difficult to directly obtain the degree of local activity during run-time. However, one can judge a resource's level of activity by studying its performance history with respect to CPU utilisation. Generally speaking, if one observes unstable oscillation (disorder) within the CPU utilisation history, one can say that is under relatively high activity and vice versa. Therefore, as a measurement of the system's degree of disorder, entropy is utilised to provide a quantitative measurement for the degree of local activity that is associated with the resource's performance.

D. Control

Controlling is an important management step. It is related to taking control of complexity. Complexity not only has to be reduced, but it also has to be avoided so that its existence in the future can be prevented. Therefore, the controlling step is made up of two parts: reducing and avoiding.

It is not always easy to completely remove complexity from the system. Thus, it is reduced as much as possible. Reducing the complexity is a strategy based on cost for realising cloud service resource management. Improving the sharing of information between the cloud consumer and provider can help lessen the high complexity and help reduce costs. However, an efficient complexity management system aims not only to reduce the complexity level by performing corrective actions, but also to avoid complexity in the future by taking preventive actions. Hence, the efficient and effective utilisation of resource analysis methods and monitor tools can help control complexity in resource management.

Algorithm 1: Calculate Resource Entropy.

Require: $CUV \leftarrow$ CPU Utilization Vector of resource
procedure: CALCULATEENTROPY*(CUV)*
  $\Delta_{\mathrm{cu}} V \leftarrow$ Vector for changes of CPU Utilization
  $Mean(\Delta_{\mathrm{cu}}) \leftarrow$ Average Changes of CPU Utilization

  if $\Delta_{\mathrm{cu}} \geq Mean(\Delta_{\mathrm{cu}})$ **then**
    $State_a \leftarrow$ Above average state
  else $State_b \leftarrow$ Below average state

  $P_a \leftarrow$ Probability of Δ_{cu} in $State_a$
  $P_b \leftarrow$ Probability of Δ_{cu} in $State_b$
  Entropy $H(\Delta_{\mathrm{cu}}) = -(P_a * log_2 P_a + P_b * log_2 P_b)$

Increasing the activity on local resources will also increase the complexity of the global resource management system. This means that system has greater probability of falling into chaos. Therefore, we are proposing the following solution in order to control the complexity, as seen in Fig. 2:

"Allocating tasks to the resources that possess or exhibit a high degree of local activity should be avoided or tasks should be allocated to the set of resources having a similar degree of local activity."

IV. APPLICATIONS AND EVALUATIONS

Based on the proposed entropy measurement, this section examines various cloud resource management strategies and provides a detailed explanation of the experiment results.

A. Resource Entropy-Based Local Activity Ranking

This paper emphasises on entropy value calculation, which is based on the resource CPU utilisation history. This gives an estimate as to how efficiently the CPU is used by the resource during executions of jobs. Since this can be directly related with the performance of the resource throughout the runtime, it becomes highly significant in making scheduling decision. Algorithm 1 is employed to calculate the resource entropy.

The following relationship is signified by the entropy measurement with the degree of resource local activity:

1) Since $0 \leq P_a, P_b \leq 1.$, entropy can be considered a non-negative quantity $H(\triangle_{\mathrm{cu}}) \geq 0$. The resource entropy value is proportional to the degree of resource local activity.
2) The maximum value of $(H(\triangle_{\mathrm{cu}}) = log_2(2) = 1)$ is achieved by entropy on occurrence of both $State_a$ and $State_b$ having the same probability $(P_a = P_b = 1/2)$. The performance of resource is determined in the most unpredictable and uncertain region, which signifies the maximum degree of activity of local resource.
3) The minimum value $H(\triangle_{\mathrm{cu}}) = 0$ is achieved by entropy with the occurrence of only one state having probability 1 ($P_a = 1$ or $P_b = 1$). Thus, the resource performance can be determined due to complete certainty, resulting in minimum degree of resource local activity.

Fig. 2. Complexity control: resource entropy based local activity ranking.

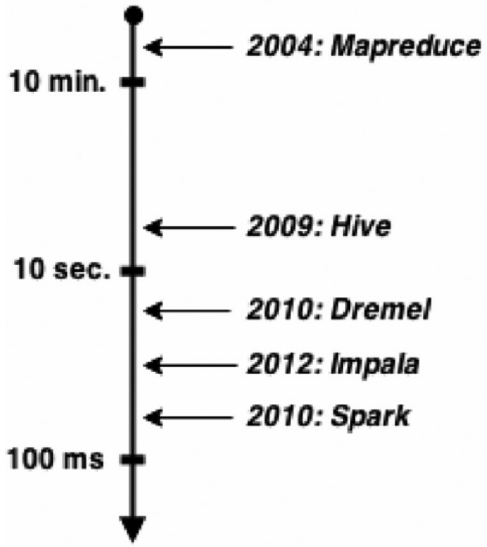


Fig. 3. Modern engines can run cloud analysis services with ever lower latency.

B. Spark Entropy Scheduler: Resource Management by Local Activity Ranking

As shown in the Fig. 3, engines such as Map Reduce [20], Hive [21], Impala [22], Dremel [23] and Spark [24] help execute cloud analysis in short time across thousands of resources. This is due to the efforts in research and industry alike, which is driven by the demand for lower-latency distributed data analysis. 3. Apache Spark, within the Apache Software Foundation, boasts of speed-ups almost 100x faster than that with Hadoop MapReduce in-memory, or 10x faster on disk. Powerful new applications such as Cloud **Analysis as A Service** [25] have been developed to bring response times into sub-second range. Apache Spark employs HTTP web service to provide cloud analysis query response/requests and support multi-threaded querying as well. A flow chart showing sending of an HTTP request is presented in Fig. 4. A thread is allocated by Spark Web Server to route the HTTP request for a specific cloud analysis job. A long run global Spark is employed for processing context jobs. The Spark Master allows scheduling to run on the pre-specified amount of Slayer Workers. In such case, sophisticated parallel

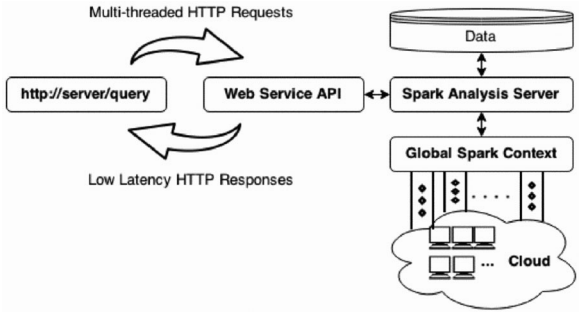


Fig. 4. Apache Spark: running Analysis as a Service (AaaS) in the Cloud.

computation, such as highly search personalisations, language translation, context recommendation and voice reorganisation can be run by employing user-facing services on a per-query basis. However, the performance of Spark decreases when faced with high concurrent of service query. Its performance can be closely linked with its resource management strategy. In most cases, increased service query requires deployment of additional resources, which also increases the underlying system's complexity.

1) Resource Management Challenge in Apache Spark: Multi-threading is supported by the Spark Context and FAIR and FIFO scheduling options for concurrent queries are also provided. Typically, multiple parallel jobs can be processed by the FAIR scheduler simultaneously for reducing overall latency. The FAIR scheduler allows assigning resources to queries so that an equal share of resources can be allocated to all queries over time on average. Fairness decisions are made by the scheduler only on the number of computing cores and memory with default setting. It assigns tasks to the resource by following random selection. CPU utilisations of the resource and core speed are not considered by FAIR scheduler, which cast a significant impact on the task's completing time. Thus, guaranteeing QoS for the on-line query is an uphill task. Popularising the web service becomes difficult should the resource management strategy fail to provide an optimal way for guaranteeing the quality of service.

It is difficult to schedule low-latency cloud analysis jobs due to multifaceted problems arising on the heterogeneous cloud. Even though Spark engines are designed for the Cloud, for

high concurrent tasks running on the heterogeneous cloud environment, they are unable to address the problem of resource scheduling. The performance of Spark can be closely linked with its job scheduler that assumes cloud resources to be homogeneous. Since the performance of resource does not change during run-time, these assumptions can be employed to make decisions on allocating jobs to resources. In practice, the resource's performance is highly dynamic in nature and assumptions of homogeneity do not always apply. Although in a homogeneous environment, the current scheduler works well, we have demonstrated that severe performance degradation occurs on breaking its underlying assumptions. Resource's performance with potentially uncontrollable variance results in server collapse on dealing with high concurrent requests. Moreover, as organisations frequently employ multiple generations of hardware to build their private cloud, we assume heterogeneous environments to become the common case.

2) Entropy Scheduler: A More Reliable and Efficient Solution: The following optimised resource management should be kept in mind:

1) The individual resource's characteristics and activities
2) The information reliability of the resource

Awareness about resource characteristics is needed for a good resource management solution. In the heterogeneous cloud, the performance of the system becomes more sensitive to resources at hand, and performance degradation can result from poor management. However, only resource's static characteristics, such as number of available cores, are considered by the native Spark Fair scheduler while overlooking dynamic characteristics such as CPU core performance. In such cases, unfair scheduling of jobs on the cores occurs with varied performance. This has a high impact on the jobs' completion time as well as predictability of system performance. Resource entropy level (REL) and resource activity vector (RAV) are introduced to capture the relevant dynamic performance characteristics of CPU core. We focus on CPU utilisation, the most important part of resource information, in the current implementation. This shows how efficiently the CPU is utilised by the operator thread during job execution. This is significant in making scheduling decision since it can be directly linked with the performance of the core during runtime. A resource monitor is run on each worker node to get RAV values. The CPU utilisation by the worker is captured by the resource monitor, and every second, RAV is updated with the CPU utilisation difference. The average change of CPU utilisation (Avg) is then calculated for each time period, followed by classification of the resource's history status into two (below average and above average). Based on algorithm 1, the REL is updated on every heartbeat interval. Then the heartbeat from the worker node is transferred to the master node with the current entropy level and CPU utilisation value to help in making jobs scheduling decision.

Spark assumes all resource to be homogeneous in nature and assigns cores to tasks randomly under Fair Scheduler. However, resources that have homogeneous setting will always function under heterogeneous performance throughout the runtime, even in the homogeneous cloud. In heterogeneous cloud, such assumptions lead to poor job completion and deliver unreliable cloud performance because of the following reasons:

1) The finish time of its slowest task determines the job completion time.
2) The chance of allocating cores with various performance levels for tasks inside a single job increases with random cores allocation.
3) The current running job has to be completed to release cores for scheduling other jobs. The other cores' computing power, those that have completed tasks, is wasted due to waiting by a job for completion of its slowest task.
4) Re-scheduling and monitoring slow tasks (conducting speculative execution of tasks) is costly.

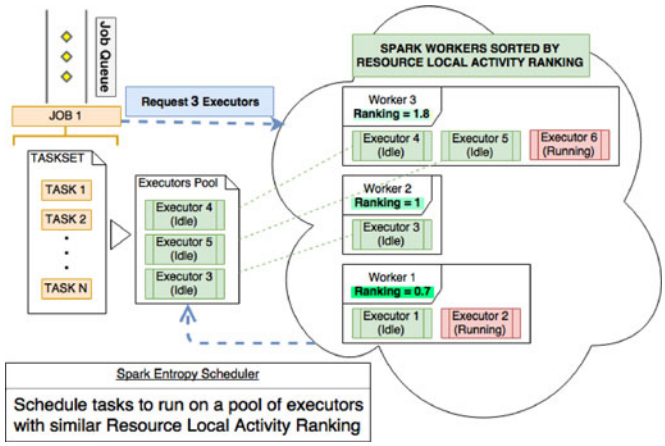


Fig. 5. Entropy Scheduler: resources allocation and tasks scheduling.

Algorithm 2: Calculate Resource Local Activity Ranking.

Require: $R_{\text{cu}} \leftarrow$ Current Resource CPU Utilization
Require: $R_e \leftarrow$ Resource Entropy
Require: $N_{\text{cpu}} \leftarrow$ Number of Available CPU cores
Require: $S_{\text{cpu}} \leftarrow$ CPU Core Clock Speed
procedure: CALCULATERANKING $R_{\text{cu}}, R_e, N_{\text{cpu}}, S_{\text{cpu}}$
  $RANK_{\text{resource}} \leftarrow$ Resource Local Activity Ranking
  $RANK_{\text{resource}} = N_{\text{cpu}} * S_{\text{cpu}} * (1 - R_{\text{cu}}) * (1 - R_e)$

In our proposed Entropy Scheduler, the resource local activity ranking of all available workers (Algorithm 2) is calculated first and the workers are sorted by the ranking thereafter. We assume that the worker is deployed on a server with same type of CPU processors. Each worker may contains one or more executors and an executor is allocated with one CPU core by default. A typical Spark job may contains more then one tasks and require one or more executors to run. Unlike the default Spark Fair Scheduler (Executors are randomly selected and allocated to a job), Entropy Scheduler pick up executors with similar Resource Local Activity Ranking to enhanced the reliability of performance and overall QoS satisfaction. Once a pool of executors are allocated to a job, the tasks are scheduled to run on the executors in "Round Robin" fashion, as shown in Fig. 5.

TABLE I
EXPERIMENTAL PLATFORM:RESOURCE SPECIFICATION

	Node 1	Node 2	Node 3
CPU	Xeon 3 Ghz x 2	Xeon 2.8 Ghz x 2	Xeon 1.8 Ghz
Cores	8	8	4
RAM	16 GB	12 GB	12 GB
Workers	2	2	1
Executors	8	8	4

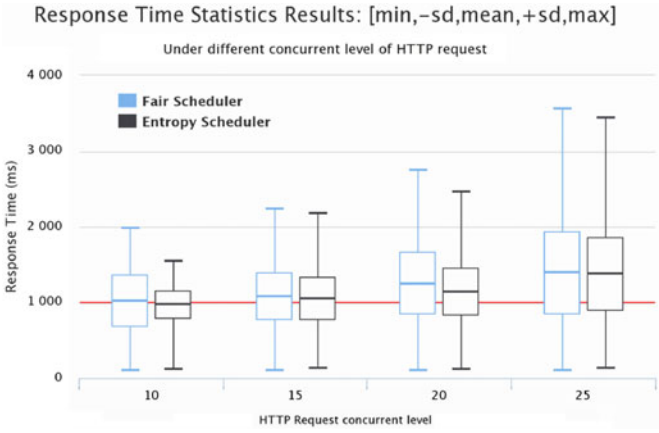


Fig. 6. Experiment 1: Statistics result for service response time.

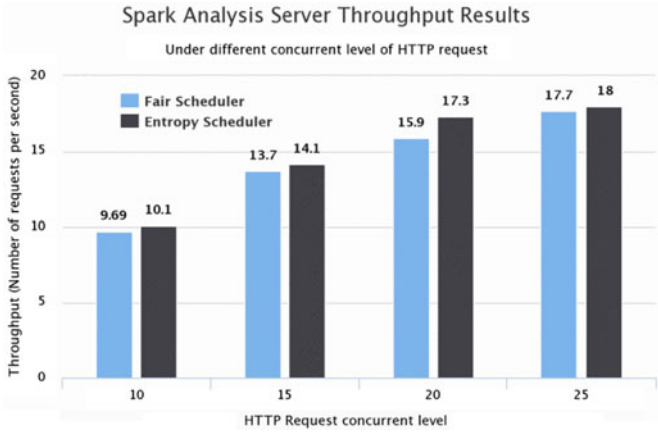


Fig. 7. Experiment 1: Overall cloud server throughput.

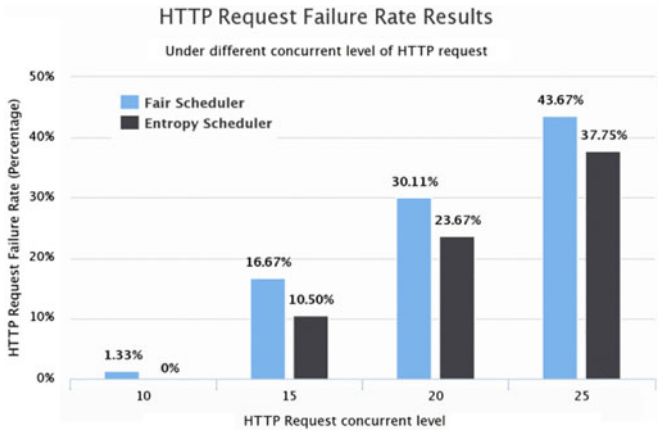


Fig. 8. Experiment 1: HTTP request failure rate result.

C. Empirical Evaluation of Entropy Scheduler

Experiments on a private cloud containing 3 physical resources with heterogeneous setting are performed to examine the proposed Entropy Scheduler. Table I presents Spark configuration and resource specifications. On the server, a simple Spark application is deployed to allow accepting user query for π calculated with a number of concurrent CPU cores that are predefined. Apache Bench is then employed for load testing the Spark application within different schedulers (Spark Fair Scheduler [26]) and Entropy Scheduler [27]). The load testing results in producing a number of threads to simultaneously execute the same query. Each thread is loaded and queries are processed by all threads until the task is completed. For performance comparison, we use the query response time of every request from all threads.

1) Experiment 1: Evaluation Under Different Concurrent Level of HTTP Request Workload: In this experiment, we validate the degree of satisfying of QoS requirement and query response time with Fair Scheduler and Entropy Scheduler under various concurrent levels of request workload, where the concurrent level refers to the number of concurrent clients trying to send the HTTP requests to the server in a given period of time (second). Figs. 6–8 show the results.

As seen in Fig. 6, Entropy Scheduler displays a higher degree of satisfying QoS requirement and better performance. This results in enhancement of the overall throughput of the server as well (Fig. 7).

However, the scheduling system faces serious challenges due to increasing workload concurrency, which also leads to degradation of cloud experience performance. Such unstable performance can be explained through two reasons:

1) Contention and load interaction amongst concurrently executing queries result in loss in stability and performance of the cloud. These effects deteriorate with more complex workloads.
2) The cloud, due to its heterogeneity and parallelism, is a difficult target to accomplish low-latency service response as performance penalties is led by poor scheduling and/or deployments.

Fig. 8 shows same performance bottlenecks inhibiting sub-second service response time even though a significant amount of failed requests is reduced by Entropy Scheduler than with Fair Scheduler. This provides motivation for other optimisation options in future work.

2) Experiment 2: Load Testing With 100,000 Service Requests at the Concurrent Level of 10: Different aspects of load testing result compared by each scheduler are presented in Table II. Throughout the Evaluation section, our results show that native Fair Scheduler is outperformed by Entropy Scheduler in terms of QoS satisfaction. On an average, Entropy Scheduler was able to reduce the load testing finish time and average service response time by almost 23% and standard deviation

TABLE II
EXPERIMENT 2: LOAD TESTING WITH 100,000 SERVICE REQUESTS AT THE CONCURRENT LEVEL OF 10

Load Testing Result	Fair Scheduler	Entropy Scheduler
Completion Time (Sec.)	951.52	732.15 (−23%)
Throughput (Queries/Sec.)	10.51	13.66 (+30%)
Number of failed request	75	0
Mean Response Time (ms)	951	732 (−23%)
Standard Deviation	298.9	194.7 (−35%)

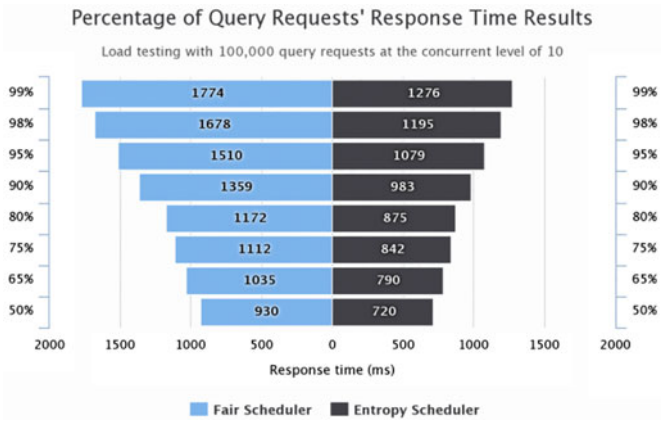


Fig. 9. Experiment 2: Percentage of the service requests served within a certain time (Million Seconds).

by 35%, in this heterogeneous cloud experiment. The overall server throughput was found to improve by almost 30% when compared with native Fair Scheduler.

According to Fig. 9, within 1 second, 90% of queries are completed under Entropy Scheduler, while under Fair Scheduler, it was only 50%. Such result supports that Entropy Scheduler is better in running cloud AaaS and provides web service with quality of service guarantee.

V. RELATED WORK

Resource Management in the Cloud has been a common area of research for many communities over the past years. However, much of the past work in research do not consider the complexity nature of cloud environment and all the solution in industry treat the Cloud environment to be simplify.

A. Resource Management in the Cloud

The fundamental kind of resource management found from existing literature can be mainly categorized into QoS based, resource based, bargaining based, prediction based, and nature-inspired/bio-inspired based.

1) QoS (e.g. Budget, Deadline, Reliability) Based: Isard et al. formulates resource assignment as a graph optimization problem, accounting for fairness, and placement constraints application may have [28]. A formulation that supports a mix of QoS scenarios with precisely defined objective function, promotes performance, fairness, and CPU utilization is proposed for static workloads with multiple types of resources by Stillwell *et al.* [29]. Byun *et al.* propose an architecture to automatically execute large-scale workflow-based applications on dynamically and elastically provisioned cloud resources [30]. Sharma *et al.* present a cost-aware resource allocation system that optimize the selection of virtual server configuration to minimizes the cost [31]. Hwang and Kim propose a cost-effective resource provisioning methodology for deadline constrained cloud applications [32]. A approach that operates fine-gained resource level scaling as well as VM level scaling (CPUs, Memory, I/O) is proposed to support cost-effective elasticity for cloud services by Han *et al.* [33]. Mao and Humphrey present an approach to ensure all jobs are finished within deadlines at lowest financial cost, where takes the virtual machine of various sizes/costs as the basic computing units and which (soft) deadlines of jobs can be specified according to the performance requirements [34]. A deadline-driven resource provision mechanism was presented to support QoS-aware execution of scientific workloads in heterogeneous cloud environment by Vecchiola *et al.* [35]. Malawski *et al.* address a resource management problem concerning IaaS project with cost budget and deadline constraints [36]. The problem of minimizing the cloud operation cost by maximizing its energy efficiency while ensuring the application's QoS requirements is addresses by Gao *et al.* later [37]. Yang *et al.* apply a dynamic interference sensitivity detection methodology to preserve the performance of batch-analysis applications for collocation scenarios [38]. Han *et al.* try to reduces the costs incurred by cloud users that using IaaS by utilizing adaptive scaling algorithms for cloud resources, which enable them to scale their applications only meets bottleneck [39]. Singh and Chana categorize the cloud application workload on the basis of common patterns and then allocating the resource according to the generalized patterns before actual scheduling [40].

2) Resource Based: A theoretical problem formulation is developed for allocating multiple heterogeneous types of resources to competing cloud services and the proposed algorithms are compared through simulation experiments based on the Google Cluster Workload [41]. Xiao, Song and Chen introduce a new concept, "Skewness", to measure the unevenness in the multi-dimensional cloud resource utilization [42]. They proposed a system to combine different types of workloads and improve the overall cloud resource utilization by minimizing the Skewness [43]. Klein *et al.* introduce Brownout that using a self-adaptation programming paradigm based on Control Theory to develop applications that can robustly withstand unpredictable resource performance without over-provisioning [44].

3) Bargaining Based: Lai *et al.* develop a cloud resource allocation system based on bargaining, which allows applications to differentiate the values of its jobs [45]. While An *et al.* propose an alternative approach where applications are allowed to automatically negotiate resource leasing contracts with cloud providers [46]. Similarly, Dastjerdi and Buyya propose a solution to automate the negotiation process in cloud environment [47]. Zhang, Zhu and Boutaba try to address the question how to best match applications QoS requirement in order to maximize cloud provider revenue and cloud users satisfactions while minimizing energy cost in a single cloud provider scenario [48]. Zaman and Grosu attempt to formulate

the problem of resource allocation in clouds as a on-line auction problem [49].

4) Prediction Based: A resource allocation methodology is presented by Gmach *et al.*, which relies on the ability to predict the cloud application's behaviour a priori [50] while Gong, Gu and Wilkes propose an alternative schema based on predictions of dynamic cloud resource run-time performance [51]. Watson *et al.* study the probabilistic relationships between resource and application and apply basic laws of probability to their proposed model to investigate whether and how CPU utilization affects application performance [52]. Shen *et al.* use on-line workload demand prediction without a priori assumptions on application behaviour to identifies the application's resource requirement, which attempt to avoid over-provisioning or over-loading of cloud resources [53]. An algorithm is proposed by Li *et al.* to adjust the number of resource allocated to applications based on the updated information of their actual task executions [54]. Islam *et al.* present a new resource measurement and provisioning solution based on prediction using Neural Network and Linear Regression to meet future workload demands [55] while Vasic *et al.* serves a similar goal by classifying workload and reuses previous resource allocations decisions to minimize reallocation overheads [56]. In Jiang *et al.* work, they attempt to make a trade-off between resource demand and service latency by automatically predict the number of application query requests [57].

5) Nature-Inspired/Bio-Inspired Based: Hegazy use the Genetic Algorithms (GAs) technique to search for near-optimum solution by taking both resource allocation and leveling heuristics into consideration [58] [59]. Hua, Zheng and Hu proposed an Ant Colony Optimization (ACO) based resource allocation algorithm to satisfy the property of cloud computing [60]. A novel parallel Q-learning approach is presented by Barrett, Howley and Duggan to reduce the overhead introduced by determine optimal policies while learning on-line [61]. Recently, a self-tuning fuzzy control (STFC) approach is extended to enable qualitative specification of elasticity rules for applications running on the cloud [62].

B. Summary

To make optimal resource management, we need to take the complex cloud resources into account. However, the lack of information regarding the dynamic cloud resources makes this problem more challenging. Nowadays, the challenges of resource management like complexity of resources (e.g. heterogeneity, dynamicity and uncertainty) are not resolved with traditional ways in cloud environment. Thus, there is a need to make cloud applications efficient by taking care of these properties of the cloud environment.

VI. Conclusion and Future Research Directions

The complexity is an important issue that affects QoS satisfaction bringing additional challenges to Cloud Service Resource Management System problem. In this paper the negative impact of complexity was used to motivate the new resource management strategy development based on **Entropy Theory**. With the results in this paper, we provide both a concrete solution for a class of complex systems, as well as a number of ideas valuable for conventional engines running on the cloud.

Complexity research is involved in a main part of 21st century science according to several prominent authors, including Stephen Hawking. However, research on Complexity has just emerged in the area of cloud resource management. The understanding of the origin of complexity (Locally-active cloud resource) and the impact of complexity (Performance degradation, QoS guarantees violation and potential Chaotic behaviour) would offer useful information to find the limitation of current resource management solutions and motivate new strategy development under complex cloud environment.

Since the approach of introducing Degree of Local Activity measured by resource entropy to control the complexity in the cloud in this paper is the first attempt in the related literature. Many problems may arise, and many issues remain open. A list of the most important ones is given in the following.

1) **New Experimentation**: The proposed ideas have to be more extensively validated in order to determine the extent to which it can improve the robust of resource management in the cloud. The validation of the ideas includes two dimensions of new experimentations:
    1) It has to be applied to more complex applications running on the cloud in order to analysis its scope and usability.
    2) It has to be applied to more complex cloud environment by involving larger amount of resources in order to analyse its scalability.

    Such experimentation is of worth interest because the final purpose is to integrate the framework in the daily practices of the resource management for cloud applications.
3) **Further Implementation**: Although the new Entropy Scheduler reduces significant amount of failure jobs compare to the native Spark Fair Scheduler, its jobs failure rate is still far from satisfaction. This problem may cause by its centralize management feature. In the future, we would like to learn the idea from other resource management systems, e.g. Apache Mesos [63], Omega [64], Sparrow [65] . . . and then transform the current solution from centralized to decentralized to solve the bottleneck problem bring by high concurrent workloads.
4) **Potential Improvement**: We assume that the resource management model only takes into account the CPU factor may usually influence by other factors as well, e.g. Memory, Disk I/O, Network . . . The model may be extended to consider these factors for potential improvement. And the current model focuses on the complexity raising from resource. In the future, complexity originated in other media (etc. links between resources, workload, outer environment) are also need to be studied.
5) **Extended Analysis**: In the current complexity management, we focus on reducing/avoiding the complexity to minimize the negative effect in the cloud resource management system. However, both positive and negative effects exist along with the increasing of complexity. There exists a completely new application of Local Activity

Principle so-called Edge of Chaos where most complex phenomena merge. The region of Edge of Chaos can mathematically rigorously be proven and confirmed with other applications in reality, which worth of extended analysis to draw on the advantages and avoid disadvantage of increasing complexity.

6) **Cross-Disciplinary Research**: Since the concept of Entropy Theory and Local Activity Principle are really fundamental in science. The concept of "Degree of Local Activity measured by Entropy" introduced in this paper may inspire future applications in other domains of computer science. For example, in intrusion detention system, Degree of Local Activity can be identified as the behaviour pattern of a user and the emerging complexity pattern generated by those locally active users may be detected as instruction. Such idea can be easily extended to other disciplinary as well, such as Weather Prediction, Road Traffic Scheduling, Calling Centre Routing. We believed our work is a step toward many fruitful research topics in the future.

REFERENCES

[1] M. Baranger, "Chaos, complexity, and entropy," New England Complex Systems Institute, Cambridge, MA, USA, 2000.

[2] F. Isik, *Complexity in Supply Chains: A New Approach to Quantitative Measurement of the Supply-Chain-Complexity*. Rijeka, Croatia: InTech, 2011.

[3] L. O. Chua, "Passivity and complexity," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 46, no. 1, pp. 71–82, Jan. 1999.

[4] L. O. Chua, "Local activity is the origin of complexity," *Int. J. Bifurcation Chaos*, vol. 15, no. 11, pp. 3435–3456, 2005.

[5] H. Chen, F. Wang, M. Migliavacca, L. O. Chua, and N. Helian, "Complexity reduction: Local activity ranking by resource entropy for QoS-aware cloud scheduling," in *Proc. IEEE Int. Conf. Serv. Comput.*, 2016, pp. 585–592.

[6] L. O. Chua, *Memristor, Hodgkin-Huxley, and Edge of Chaos*. New York, NY, USA: Springer, 2014.

[7] J. Xie, Y. Deng, G. Min, and Y. Zhou, "An incrementally scalable and cost-efficient interconnection structure for data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 6, pp. 1578–1592, Jun. 2017.

[8] Y. Deng, "What is the future of disk drives, death or rebirth?," *ACM Comput. Surveys.*, vol. 43, no. 3, 2011, Art. no. 23.

[9] A. Iosup, N. Yigitbasi, and D. Epema, "On the performance variability of production cloud services," in *Proc. 11th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput*, 2011, pp. 104–113.

[10] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, "Runtime measurements in the cloud: Observing, analyzing, and reducing variance," *Proc. VLDB Endowment*, vol. 3, no. 1/2, pp. 460–471, 2010.

[11] W. Herroelen and R. Leus, "Project scheduling under uncertainty: Survey and research potentials," *Eur. J. Oper. Res.*, vol. 165, no. 2, pp. 289–306, 2005.

[12] H. Chen, F. Wang, and N. Helian, "A cost-efficient and reliable resource allocation model based on cellular automaton entropy for cloud project scheduling," *System*, vol. 4, no. 4, pp. 7–14, 2013.

[13] L. Boltzmann, "The second law of thermodynamics," in *Theoretical Physics and Philosophical Problems*. New York, NY, USA: Springer, 1974, pp. 13–32.

[14] S. Christodoulou, G. Ellinas, and P. Aslani, "Entropy-based scheduling of resource-constrained construction projects," *Autom. Construction*, vol. 18, no. 7, pp. 919–928, 2009.

[15] F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller, and J. Lawall, "Entropy: A consolidation manager for clusters," in *Proc. ACM SIGPLAN/SIGOPS Int. Conf. Virtual Execution Environ.*, 2009, pp. 41–50.

[16] H.-S. Gan and A. Wirth, "Comparing deterministic, robust and online scheduling using entropy," *Int. J. Prod. Res.*, vol. 43, no. 10, pp. 2113–2134, 2005.

[17] J. Liu, H. Tu, H. Zhang, F. Xia, and D. Yu, "Research on measurement entropy-based of equipment management complexity and its application in production planning," in *Intelligent Robotics and Applications*. New York, NY, USA: Springer, 2008, pp. 604–611.

[18] C. G. Langton, "Computation at the edge of chaos: Phase transitions and emergent computation," *Phys. D, Nonlinear Phenom.*, vol. 42, no. 1, pp. 12–37, 1990.

[19] R. A. Silverman and M. D. Friedman, *Mathematical Foundations of Information Theory*. New York, NY, USA: Dover, 1957, vol. 434.

[20] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[21] A. Thusoo *et al.*, "Hive: A warehousing solution over a map-reduce framework," *Proc. VLDB Endowment*, vol. 2, no. 2, pp. 1626–1629, 2009.

[22] M. Kornacker *et al.*, "Impala: A modern, open-source SQL engine for Hadoop," in presented at the 7th Biennial Conf. Innov. Data Syst. Res., Asilomar, CA, USA, Jan. 4–7, 2015.

[23] S. Melnik *et al.*, "Dremel: Interactive analysis of web-scale datasets," *Proc. VLDB Endowment*, vol. 3, no. 1/2, pp. 330–339, 2010.

[24] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," *HotCloud*, vol. 10, pp. 10–10, 2010.

[25] D. Xu, D. Wu, X. Xu, L. Zhu, and L. Bass, "Making real time data analytics available as a service," in *Proc. 11th Int. ACM SIGSOFT Conf. Qual. Softw. Archit.*, 2015, pp. 73–82.

[26] M. Zaharia, "Job scheduling with the fair and capacity schedulers," in *Proc. Hadoop Summit*, 2009, vol. 9.

[27] H. Chen and F. Z. Wang, "Spark on entropy: A reliable & efficient scheduler for low-latency parallel jobs in heterogeneous cloud," in *Proc. IEEE 40th Local Comput. Netw. Conf. Workshops*, 2015, pp. 708–713.

[28] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: Fair scheduling for distributed computing clusters," in *Proc. ACM SIGOPS 22nd Symp. Oper. Syst. Princ.*, 2009, pp. 261–276.

[29] M. Stillwell, D. Schanzenbach, F. Vivien, and H. Casanova, "Resource allocation algorithms for virtualized service hosting platforms," *J. Parallel Distrib. Comput.*, vol. 70, no. 9, pp. 962–974, 2010.

[30] E.-K. Byun, Y.-S. Kee, J.-S. Kim, and S. Maeng, "Cost optimized provisioning of elastic resources for application workflows," *Future Gener. Comput. Syst.*, vol. 27, no. 8, pp. 1011–1026, 2011.

[31] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A cost-aware elasticity provisioning system for the cloud," in *Proc. 31st Int. Conf. Distrib. Comput. Syst.*, 2011, pp. 559–570.

[32] E. Hwang and K. H. Kim, "Minimizing cost of virtual machines for deadline-constrained mapreduce applications in the cloud," in *Proc. ACM/IEEE 13th Int. Conf. Grid Comput.*, 2012, pp. 130–138.

[33] R. Han, L. Guo, M. M. Ghanem, and Y. Guo, "Lightweight resource scaling for cloud applications," in *Proc. 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, 2012, pp. 644–651.

[34] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2011, Art. no. 49.

[35] C. Vecchiola, R. N. Calheiros, D. Karunamoorthy, and R. Buyya, "Deadline-driven provisioning of resources for scientific applications in hybrid clouds with Aneka," *Future Gener. Comput. Syst.*, vol. 28, no. 1, pp. 58–65, 2012.

[36] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2012, Art. no. 22.

[37] Y. Gao, Y. Wang, S. K. Gupta, and M. Pedram, "An energy and deadline aware resource provisioning, scheduling and optimization framework for cloud systems," in *Proc. 9th IEEE/ACM/IFIP Int. Conf. Hardware/Softw. Codesign Syst. Synthesis*, 2013, Art. no. 31.

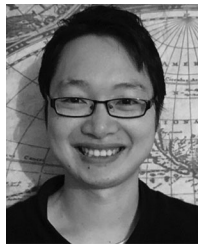
[38] H. Yang, A. Breslow, J. Mars, and L. Tang, "Bubble-flux: Precise online QoS management for increased utilization in warehouse scale computers," *ACM SIGARCH Comput. Archit. News*, vol. 41, no. 3, pp. 607–618, 2013.

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

[40] S. Singh and I. Chana, "Q-aware: Quality of service based cloud resource provisioning," *Comput. Elect. Eng*, vol. 47, pp. 138–160, 2015.

[41] M. Stillwell, F. Vivien, and H. Casanova, "Virtual machine resource allocation for service hosting on heterogeneous distributed platforms," in *Proc. IEEE 26th Int. Parallel Distrib. Process. Symp.*, 2012, pp. 786–797.

[42] J. Mars and L. Tang, "Whare-map: Heterogeneity in homogeneous warehouse-scale computers," *ACM SIGARCH Comput. Archit. News*, vol. 41, no. 3, pp. 619–630, 2013.
[43] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1107–1117, Jun. 2013.
[44] C. Klein, M. Maggio, K.-E. Årzén, and F. Hernández-Rodriguez, "Brownout: Building more robust cloud applications," in *Proc. 36th Int. Conf. Softw. Eng.*, 2014, pp. 700–711.
[45] K. Lai, L. Rasmusson, E. Adar, L. Zhang, and B. A. Huberman, "Tycoon: An implementation of a distributed, market-based resource allocation system," *Multiagent Grid Syst.*, vol. 1, no. 3, pp. 169–182, 2005.
[46] B. An, V. Lesser, D. Irwin, and M. Zink, "Automated negotiation with decommitment for dynamic resource allocation in cloud computing," in *Proc. 9th Int. Conf. Autonomous Agents Multiagent Syst.*, 2010, vol. 1, pp. 981–988.
[47] A. V. Dastjerdi and R. Buyya, "An autonomous reliability-aware negotiation strategy for cloud computing environments," in *Proc. 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, 2012, pp. 284–291.
[48] Q. Zhang, Q. Zhu, and R. Boutaba, "Dynamic resource allocation for spot markets in cloud computing environments," in *Proc. 4th IEEE Int. Conf. Utility Cloud Comput.*, 2011, pp. 178–185.
[49] S. Zaman and D. Grosu, "Combinatorial auction-based allocation of virtual machine instances in clouds," *J. Parallel Distrib. Comput.*, vol. 73, no. 4, pp. 495–508, 2013.
[50] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload analysis and demand prediction of enterprise data center applications," in *Proc. IEEE 10th Int. Symp. Workload Characterization*, 2007, pp. 171–180.
[51] Z. Gong, X. Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems," in *Proc. Int. Conf. Netw. Serv. Manage.*, 2010, pp. 9–16.
[52] B. J. Watson, M. Marwah, D. Gmach, Y. Chen, M. Arlitt, and Z. Wang, "Probabilistic performance modeling of virtualized resource allocation," in *Proc. 7th Int. Conf. Autonomic Comput.*, 2010, pp. 99–108.
[53] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: Elastic resource scaling for multi-tenant cloud systems," in *Proc. 2nd ACM Symp. Cloud Comput.*, 2011, Art. no. 5.
[54] J. Li, M. Qiu, Z. Ming, G. Quan, X. Qin, and Z. Gu, "Online optimization for scheduling preemptable tasks on IaaS cloud systems," *J. Parallel Distrib. Comput.*, vol. 72, no. 5, pp. 666–677, 2012.
[55] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Gener. Comput. Syst.*, vol. 28, no. 1, pp. 155–162, 2012.
[56] N. Vasic, D. Novakovic, D. Kostic, S. Miucin, and R. Bianchini, "Accelerating resource allocation in virtualized environments using workload classes and/or workload signatures," U.S. Patent 13/411 491, Mar. 2, 2012.
[57] J. Jiang, J. Lu, G. Zhang, and G. Long, "Optimal cloud resource auto-scaling for web applications," in *Proc. 13th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, 2013, pp. 58–65.
[58] T. Hegazy, "Optimization of resource allocation and leveling using genetic algorithms," *J. Construction Eng. Manage.*, vol. 125, no. 3, pp. 167–175, 1999.
[59] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in *Proc. 24th IEEE Int. Conf. Adv. Inf. Netw. Appl.*, 2010, pp. 400–407.
[60] X.-y. Hua, J. Zheng, and W.-x. Hu, "Ant colony optimization algorithm for computing resource allocation based on cloud computing environment," *J. East China Normal Univ.*, vol. 1, no. 1, pp. 127–134, 2010.
[61] E. Barrett, E. Howley, and J. Duggan, "Applying reinforcement learning towards automating resource allocation and application scalability in the cloud," *Concurrency Comput., Pract. Exp.*, vol. 25, no. 12, pp. 1656–1674, 2013.
[62] J. Rao, Y. Wei, J. Gong, and C.-Z. Xu, "QoS guarantees and service differentiation for dynamic cloud applications," *IEEE Trans. Netw. Serv. Manage.*, vol. 10, no. 1, pp. 43–55, Mar. 2013.
[63] B. Hindman *et al.*, "Mesos: A platform for fine-grained resource sharing in the data center," in *Proc. 8th USENIX Conf. Netw. syst. Des. Implementation*, 2011, vol. 11, pp. 22–22.
[64] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, and J. Wilkes, "Omega: Flexible, scalable schedulers for large compute clusters," in *Proc. 8th ACM Eur. Conf. Comput. Syst.*, 2013, pp. 351–364.
[65] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica, "Sparrow: Distributed, low latency scheduling," in *Proc. 24th ACM Symp. Oper. Syst. Princ.*, 2013, pp. 69–84.



Huankai Chen (S'13) received the M.Sc. degree in computer science from the University of Hertfordshire, Hatfield, U.K., in 2009, and the PGDip degree in actuarial science from the University of Southampton, Southampton, U.K., in 2011. He is currently working toward the Ph.D. degree in computer science at the University of Kent, Canterbury, U.K. His research interests include cloud computing, big data, and intelligent information processing.

Frank Z. Wang received the Ph.D. degree in U.K. in 1999. He is a Professor in the Future Computing of School of Computing, University of Kent, Canterbury, U.K. He was the Head of School of Computing at University of Kent for six years. He has been invited to deliver keynote speeches and invited talks to report his research worldwide, for example, at Princeton University, Carnegie Mellon University, CERN, University of Technology Sydney, Hong Kong University of Science and Technology, Tsinghua University (Taiwan), Jawaharlal Nehru University, Aristotle University, and University of Johannesburg. In 1996, he designed and developed spin-tunneling random access memory at Tohoku University, Japan, which was the first of its kind worldwide. In 2004, he was appointed as the Chair & Professor, the Director of Centre for Grid Computing at Cambridge-Cranfield High Performance Computing Facility (CCHPCF). CCHPCF is a collaborative research facility in the Universities of Cambridge and Cranfield (with an investment size of 40 million). His research interests include memristor for future computing, neuromorphic architecture, brain-like computer, chaotic behavior of cloud computing, big data, bioinspired computing, and green computing. He and his team have developed grid-oriented storage with the sponsorship of EPSRC/DTI and received the ACM/IEEE Super Computing Finalist Award. He is the Chairman (UK & Republic of Ireland Chapter) of the IEEE Computer Society and a Fellow of the British Computer Society. He has served the UK Government EPSRC e-Science Panel and the Irish Government High End Computing Panel for Science Foundation Ireland.

Na Helian received the Ph.D. degree in computer science from the Huazhong University of Science and Technology, Wuhan, China, in 1993. She has various working experiences in Japan, Singapore, and U.K. She is currently a Principal Lecturer in the School of Computer Science, University of Hertfordshire, Hatfield, U.K. Her research interests include clouding computing and data mining.