# Complexity Reduction: Local Activity Ranking By Resource Entropy For QoS-aware Cloud Scheduling

Huankai Chen*, Frank Wang*, Matteo Migliavacca*, Leon O. Chua†, Na Helian‡
*Future Computing Group, School of Computing, Canterbury, UK
{HC269, F.Z.Wang, M.Migliavacca}@kent.ac.uk
†Depart. of Electrical Engineering and Computer Science, University of California, Berkeley, USA
‡School of Computing, University of Hertfordshire, Hertfordshire, UK

***Abstract*—The principle of local activity originated from electronic circuits, but can easily translate into other non-electrical homogeneous/heterogeneous media. Cloud resource is an example of a locally-active device, which is the origin of complexity in cloud scheduling system. However, most of the researchers implicitly assume the cloud resource to be locally passive when constructing new scheduling strategies. As a result, their research solutions perform poorly in the complex cloud environment. In this paper, we first study several complexity factors caused by the locally-active cloud resource. And then we extended the "Local Activity Principle" concept with a quantitative measurement based on Entropy Theory. Furthermore, we classify the scheduling system into "Order" or "Chaos" state with simulating complexity in the cloud. Finally, we propose a new approach to controlling the chaos based on resource's Local Activity Ranking for QoS-aware cloud scheduling and implement such idea in Spark. Experiments demonstrate that our approach outperforms the native Spark Fair Scheduler with server cost reduced by 23%, average response time improved by 15% - 20% and standard deviation of response time minimized by 30% - 45%.**

***Keywords*—Local Activity Principle, Entropy Theory, Cloud Scheduling, Quality of Service, Complex System, Order and Chaos**

## I. Introduction

"Local Activity Principle" was originally used for study the complex system in physics, chemistry, biology and brain research, which is capable of explaining the emergence of complex pattern in a homogeneous medium [1]. However, the application of local activity principle in complex cloud scheduling system is limited. In cloud computing, complexity limited the system's ability to better satisfy the QoS requirements of applications, such as cost budget, average task runtime and reliability [4]. As the origin of complexity, the locally-active resource, is assumed to be locally passive in most of the research solutions. Such improper assumption may lead the scheduling solution to be less robust in the real world complex cloud environment.

Scheduling is an NP-complete problem, the complexity of which increase substantially in heterogeneous cloud environment [6]. Cloud application that disposes of scheduler, which automatically and efficiently find the most appropriate resources to execute a group of tasks, must cope with world's natural tendency to disorder. In the cloud application, jobs are scheduled on a set of cloud resources that are locally active, which performance is supposed to change dynamically during runtime [2]. Such performance diversion may cause by hardware/software failures, resources CPU overload, resource over- or under-provisioning, or application misbehaviours. We want resource local activity yield coherent global schedule system order. However, widespread experience warns us that optimizing systems that exhibit both local activity and global order are not easy. The experience that anything that can go wrong will go wrong and at the worst possible moment is summarized informally as Murphys Law [5]. Scheduling systems are not immune to Murphy. As the degree of cloud resource activity increase, the level of complexity in scheduling system increase, which may lead the system falls into the chaotic state. In chaotic state, the scheduling system performance is degraded and become harder to be predicted, and the QoS requirements of application become harder to be satisfied as well.

At the root of the ubiquity of disordering tendencies is the Second Law of Thermodynamics [3], Energy spontaneously tends to flow only from being concentrated in one place to becoming diffused or dispersed and spread out. In cloud scheduling system, adding resources to a system may overcome the Second Law spontaneous tendency and lead to increasing the systems order. However, this way does not work well all the time, especially when the cloud resources are locally active, which is the origin of complexity [1]. The scheduling system becomes more complex as more resources need to manage. In such case, the way to decide the number of resources allocated to the application initially and finding a suitable set of resources for the jobs during runtime become a critical problem in cloud scheduling. To solve the above problem, we need to know: 1) The state of cloud scheduling system, "Order","Edge of Chaos" or "Chaos", when meeting the different level of complexity with the number of allocated resource. 2) The degree of local activity for allocated resources during runtime, which has a direct impact on the system's complexity level.

The main contributions of our paper are the following:

- "Local Activity Principle" was first applied on cloud scheduling system in the literature to find the origin of complexity in cloud computing.
- We extend the concept of "Local Activity Principle" by introducing Degree of Local Activity, which can be quantitatively measured by resource Entropy for the

978-1-5090-2628-9/16 $31.00 © 2016 IEEE
DOI 10.1109/SCC.2016.82

purpose of complexity reduction and chaos control.

- We study the negative impact of complexity in cloud scheduling system through simulation, such as performance degradation and QoS guarantees violation.
- We confirm the finding of chaotic behaviour on cloud scheduling system in some complexity region and provide a way to classify the system state, "Order" or "Chaos".
- A new Entropy Scheduler was developed based on Resource Local Activity Ranking to ensure QoS guaranteed on the real world cloud analysis engine - Spark. Experiments show that our proposed Entropy Scheduler outperform the native Spark Fair Scheduler for better QoS satisfaction.

In this paper, following the short introduction on the "Local Activity Principle" [1] and the application of Entropy as the quantitative measure of the degree of cloud resource local activity, we use Damage Spreading Method [9] as a tool to analysis the simulation results provided by ComplexCloudSim in Section III. We will then describe the experiment that runs on the real world cloud analysis engine which implements our proposal idea as a plug-in scheduler and evaluates the results in Section IV. Section V contains some conclusion and possible future research direction.

## II. Locally-Active Resource : Origin of Complexity in Cloud Scheduling

The principle of local activity originated from electronic circuits, but can easily translate into other non-electrical homogeneous/heterogeneous media [1]. In cloud computing, the cloud resource is an example of a locally-active device, whereby a "small" (estimated runtime of allocated task) input signal can convert into a "large" (Actual processing time to finish the assigned task) output signal at the expense of an energy supply (cost of resource), as shown in Figure 1. By definition, a resource is locally passive if it is not locally active, in the sense that a resource with fixed cost is guaranteed to provide a never changed performance during runtime. However, in the real world cloud, the resources are seldom in the passive mode, they always exhibit the different degree of local activity. For example, on average, a physical resource is less active than a virtual resource with the same configuration and the degree of activity for the same resource varies during runtime.

### A. Complexity Caused By Locally-Active Cloud Resource

As the origin of complexity, the local activity resource has a direct impact on the complexity level of cloud scheduling system. In electronic circuits with homogeneous media, the locally active cells will put the system to be in the "Edge of Chaos" [12] state in some parameter regions, which have a chance to turn into a complete Chaotic state. In cloud environment,such complexity effects causing by locally active resource will appears more frequently. When the cloud scheduling system is in chaotic state, its performance is degraded and become harder to predict and it fails to better fulfil the QoS requirements of the application. However, in the literature, most of the researchers ignore the impacts of local activity of resource on cloud scheduling system and assume the resources to be locally passive when constructing new scheduler. So their research solution always fail to provide better QoS when running on real world cloud environment.

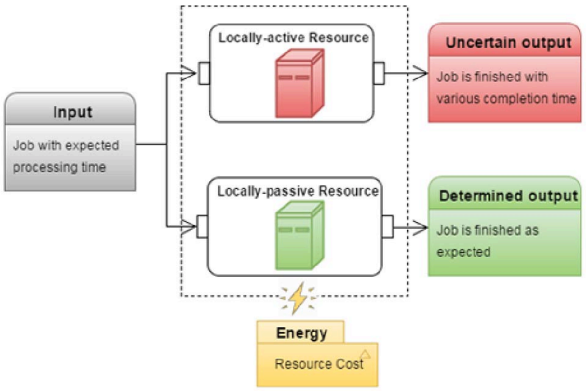


Fig. 1. Locally-Active Resource Vs. Locally-Passive Resource

The scheduling problem in cloud computing is not new at all; as a matter of fact it is one of the most studied problems in the optimization community [13] [15]. However, in the cloud the complexity causing by locally active resources that makes the problem more challenge. Some of the complexity factors related to the resource are the following:

- **Heterogeneity** : Cloud systems act as large virtual supercomputer, yet the computational resources could be very disparate, ranging from laptops, desktops, clusters, supercomputers and even small devices of limited computational resources like the smart phone. Current Cloud infrastructures are not yet much versatile but heterogeneity is among most important features to take into account in any cloud system. With the development of virtualization technology, a single physical host can run multiple virtual machines (Vms) simultaneously. Nevertheless, the virtualization also brings about new challenges to the resource scheduling in clouds since multiple VMs can share the hardware resources (e.g. CPU, memory, I/O, network, etc.) of a physical machine. In such situation, it is difficult to accurately measure the actual performance of rented VMs. For example, in Amazon EC2, the provisioning of resources to virtual machines is based on computing units instead of fixed performance measures. Different host machines provide a different amount of computing power per provisioned compute unit, effectuating in heterogeneity among VM performance [16]. That means, in real world, the cloud could never be homogeneous, it should always be heterogeneous.
- **Dynamicity** : The dynamic changes of resource performance at runtime is another important factor of complexity inherent to cloud computing [17]. In the real world scenario, such dynamicity of resource performance may be caused by hardware/software failures, resource

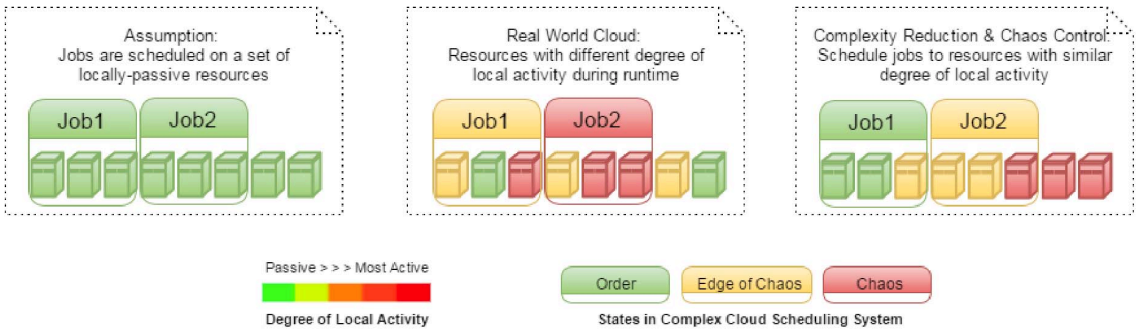


Fig. 2. Complexity Reduction & Chaos Control: Resource Entropy Based Local Activity Ranking

CPU overload, resource over- or under-provisioning, or application misbehaviours. The cloud resource is also affected by the amount of running jobs that assigned to it and exhibited local activity, which is the origin of complexity. Furthermore, sharing common underlying hardware infrastructure with other VMs will bring the resource dynamicity up to a more complex level.

- **Uncertainty** : The vast majority of the research efforts in scheduling assumes complete information about the state of cloud resource. However, in the cloud computing, the ready time and the computing capacity of a resource are subject to considerable uncertainty during provisioning [18]. We argue that such uncertainty is the main hassle of cloud computing bringing additional challenges to predict the execution time of tasks, which is a crucial point for many scheduling algorithms. Resource states in cloud environment can change dramatically. Most of the time, it is impossible to get exact knowledge about the resource. It is hard to estimate runtime of tasks accurately, improve prediction by historical data, prediction correction, prediction fallback, etc. The inaccurate execution prediction leaves the associated scheduling performance under considerable uncertainty.

### *B. Emergence Of Complex Patterns In Cloud Scheduling: Order, Edge Of Chaos And Chaos*

The principle of local activity is the cause of symmetry breaking in homogeneous media, which offers a rigorous and effective tool to identify the states (See Figure 2) of scheduling system and also fine tuning such states into a relatively small subset called the edge of chaos where the emergence of complex phenomena is most likely [1].

The increment of activity on local resource will lead to the increment of global scheduling system's complexity, which means the system will have a higher chance to fall into chaos. Thus, we propose the following solution to reduce the complexity and control the chaos, as shown in Figure 2:

"*Avoid allocate tasks to the resources with high degree of local activity* or *allocate tasks to the set of resources with similar degree of local activity* when making scheduling decision."

However, it brings up another challenging problem:

"*How to provide a quantitative measurement of resource local activity during runtime in an efficient and reliable way?*"

Therefore, to solve the problem, we introduce Entropy as the quantitative measurement to compare the degree of Local Activity among cloud resources. The aim of Local Activity measurement is to be able to obtain a numerical scale to compare the activity degree on different resources. In practical, the degree of local activity is difficult to obtain directly on runtime. However, we can judge how active a resource is through the study of its performance history in respect of CPU Utilization. General speaking, if the resource CPU Utilization history exhibit unstable oscillation (disorder), it is under relatively high activity and vice verses. Therefore, Entropy, as the measurement of the degree of disorder in a system, is used to provided a quantitative measurement of the local activity degree associated with the cloud resources.

The concept of entropy is originally known as the second law of thermodynamics, which has been adapted in other fields of study, including information theory, production planning, resource management, computer modelling and simulation. Shannon describes the entropy as a measure of information or uncertainty on random variables, which take different probabilities among the states into account [20]. The average uncertainty associated with an outcome is represented by discrete random variable $X$ on a finite set $X = x_1, ..., x_n$ with probability distribution function $p(x_i)$ being in state $i$, $(i = 1, ..., n)$. The Shannon's information entropy $H(X)$ of $X$ is defined as

$$H(X) = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i) \quad (1)$$

This paper focuses on calculating the entropy value based on the resource CPU utilization history, which represents how efficiently the resource uses the CPU throughout the jobs executions. This is highly relevant for making scheduling decision as it is directly related to the resource's performance during runtime. The resource entropy is calculated according to the algorithms 1.

The Entropy measurement above represents the following relationship with the degree of resource local activity:

**Algorithm 1** Calculate Resource Entropy

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

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

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

- Entropy is a non-negative quantity: $H(\triangle_{cu}) \geq 0$, since $0 \leq P_a, P_b \leq 1$. The degree of resource local activity is proportion to the resource entropy value.
- Entropy achieves its maximum value ($H(\triangle_{cu}) = log_2(2) = 1$) when both $State_a$ and $State_b$ occur with the same probability ($P_a = P_b = 1/2$), so the resource performance is being in most uncertain and unpredictable region, which means the degree of resource local activity is maximum.
- Entropy attains its minimum value $H(\triangle_{cu}) = 0$ when only one state occurs with probability 1 ($P_a = 1$ or $P_b = 1$), so the resource performance is known with complete certainty, then the degree of resource local activity is minimum.

## III. Order And Chaos In Complex Scheduling System

In this section, we first use ComplexCloudSim, which is an extension to popular CloudSim tool-kit with providing the capacity to model the complexity factors (Heterogeneity, Dynamicity and Uncertainty), to simulate the impacts of complexity causing by locally-active resources on the cloud scheduling system. In the simulation, we use a Montage workflow come with CloudSim, which consists of 1000 jobs with groups of random number sub tasks. For the initial simulation configuration, we set the number of VMs $Number_{vm} = 5$ and the degree of resource complexity $Degree_{complexity} = 0$. The workload will run with MinMin algorithm, which is a simple and efficient algorithm that produces a better schedule that minimizes the total completion time of jobs than other algorithms in the literature [13] [14], on the initial configuration 100 times to generate baseline performance. As what we have expected, the workflow runtime was determined in all the 100 simulation runs with zero variance without considering the complexity, which is shown in Table I.

And then, we run the simulation with the same number of VMs $Number_{vm} = 5$ but different degree of complexity $Degree_{complexity} \in [0.1, 0.2, 0.3, 0.4, 0.5]$. The results of the experiment outlined above are displayed in Figure 3 and 4. Over the course of the entire experiments, the average runtime of the Montage workflow between 3,220 and 3,424 minutes

TABLE I
BASELINE SIMULATION RESULT WITH INITIAL CONFIGURATION : $Number_{vm} = 5$,$Degree_{complexity} = 0$

| Algorithm | Average Runtime | Variance | Standard Deviation |
|---|---|---|---|
| MinMin | 2864 Minutes | 0 | 0 |

have been observed in Figure 3, which means around 13% - 23% runtime degradation compared with the performance baseline. Clearly, the complexity factors have a considerable impact on QoS of cloud scheduling system.

We also find that the average runtime degradation does not change as much as the increase of the degree of complexity. However, the growth of standard deviation for workflow runtime is proportional to the increase of the degree of complexity with range from 20% to 120%, as shown on Figure 4. Apparently, the increase of standard deviation leads to less reliable scheduling performance. Thus, the cloud scheduling QoS is depended on the degree of complexity.

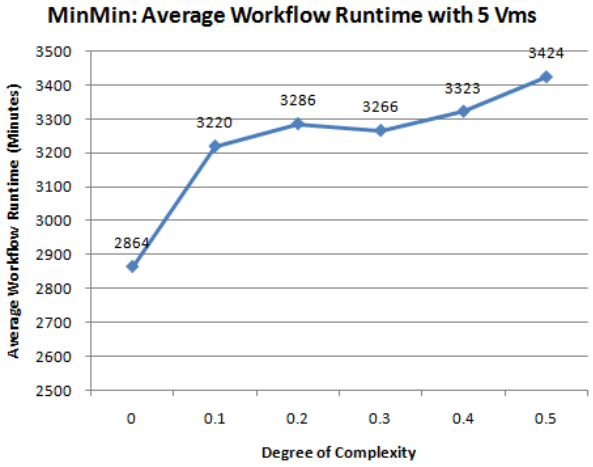


Fig. 3. Complexity Simulation: Average Workflow Runtime (MinMin,$Number_{vm} = 5$)

Finally, we introduce Damage Spreading Analysis (DSA) [8], which is a tool originally developed to study biologically motivated complex systems, and it appears in the literature on various research areas including complex network models as a way to observe the complex behaviour of the systems. DSA investigates the evolution of slightly different configuration of variables in a complex system, which are subjected to the same number sequence. Knowledge of whether or not a small perturbation (”damage” to the conditions) added to the variables spreads or stays at the same level (even disappears) can help us to investigate the robustness of the system over disturbance.

”Initial damage” here is defined as a slight change in the degree of resource complexity $C_{complexity}$ and the number of VMs $C_{vm}$ to run the same workload. We add small change $C_{complexity} = 0.1$ and $C_{vm} = 1$ to simulation step

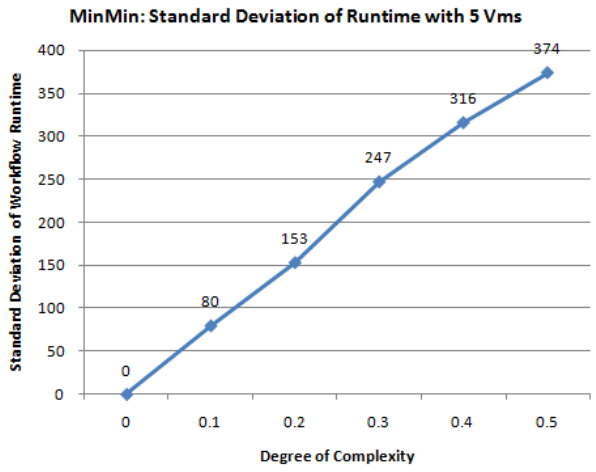


Fig. 4. Complexity Simulation: Standard Deviation of Workflow Runtime (MinMin,$Number_{vm} = 5$)

by step, which will be executed 100 times with the same workload. Then we investigate the changes are spread or not on two important QoS requirements in the scheduling processes (Changes of Average and Standard Deviation of workflow runtime) after that.

$$D_{average}(i,j) = R_{average}(i + C_{vm}, j) - R_{average}(i,j) \quad (2)$$

$$D_{std}(i,j) = R_{std}(i, j + C_{complexity}) - R_{std}(i,j) \quad (3)$$

To evaluate the spread of the damages, we define damage $D_{average}$ (Difference of average workflow runtime $R_{average}$) and $D_{std}$ (Difference of workflow runtime Standard Deviation $R_{std}$) between two simulations results, which are calculated as shown in Formula 2 and 3, where $i \in [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]$ refers to number of VMs and $j \in [0.1, 0.2, 0.3, 0.4, 0.5]$ refers to degree of complexity.

The results of $D_{average}$ and $D_{std}$ are shown in Figure 5 and 6 respectively.

As we can see from Figure 5, for number of VMs $i < 10$, the changes of $D_{average}$ for different degrees of complexity is relatively small, in this region, the damage is not spread and initial damage stays small.

From Figure 6, for number of VMs $i < 9$, the changes of $D_{std}$ for different degrees of complexity highly unstable, but the situation become relatively better as the number of VMs increase when $i > 9$.

Then,we analysis the relation between number of increased VMs $i$ and spreading damage using the standard deviation of $D_{average}$ and $D_{std}$. We define standard deviation of $D_{average}(i)$ as $\sigma_{average}(i)$, and standard deviation of $D_{std}(i)$ as $\sigma_{std}(i)$. And calculate the mean value $Mean(\sigma_{average})$ and

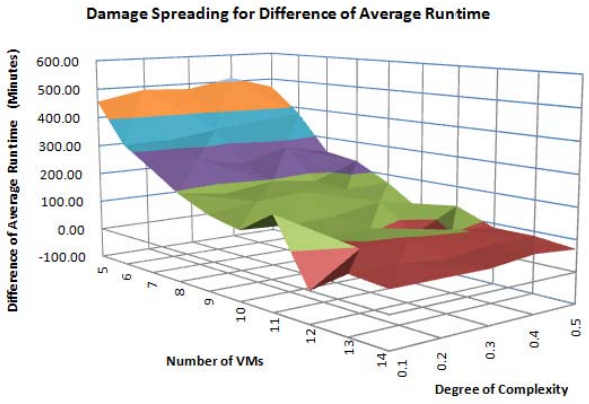


Fig. 5. Damage Spreading Evaluation: $D_{average}$

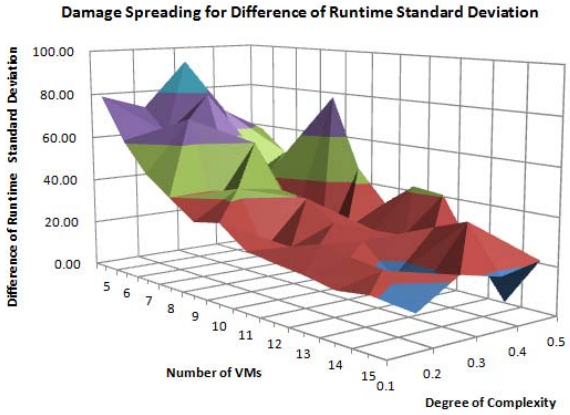


Fig. 6. Damage Spreading Evaluation: : $D_{std}$

$Mean(\sigma_{std})$ of all $\sigma_{average}$ and $\sigma_{std}$, as shown on Table II and III.

Now, we classify the system state loosely using such mean value. We understand the state that $\sigma_{average}(i) \leq Mean(\sigma_{average})$ or $\sigma_{std} \leq Mean(\sigma_{std})$ as "Order" state. In this state, the correlation of initial damage and spreading damage is maintained, the increase of number of VMs will result in steady improvement of QoS, which means the scheduling system is running relatively robust against the changes of the degree of complexity. We also understand that $\sigma_{average}(i) > Mean(\sigma_{average})$ or $\sigma_{std} > Mean(\sigma_{std})$ as "Chaos" state, as highlighted in red colour in Table II and III . In this state, small disturbance may spread throughout the scheduling system and the performance is easily changed totally against the degree of complexity, which means the increased of number of VMs is hard to grantee better QoS improvement.

The understanding of whether the scheduling system is in "Order" state or "Chaos" state provide us an important guideline for making the decision to achieve more robust scheduling. For example, from simulation result, we may run

TABLE II
RELATION BETWEEN NUMBER OF VMS AND $D_{average}$

| | $D_{average}(i)$ | | | | | |
|---|---|---|---|---|---|---|
| | Degree of Complexity | | | | | Mean($\sigma_{average}$)=23 |
| ($i$) VMs | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | $\sigma_{average}(i)$ |
| 5 | 456 | 489 | 481 | 514 | 469 | 22 |
| 6 | 320 | 322 | 344 | 363 | 377 | 25 |
| 7 | 258 | 271 | 237 | 282 | 248 | 18 |
| 8 | 193 | 174 | 196 | 178 | 231 | 23 |
| 9 | 148 | 168 | 180 | 169 | 171 | 12 |
| 10 | 124 | 117 | 122 | 149 | 94 | 19 |
| 11 | 198 | 101 | 108 | 64 | 135 | 50 |
| 12 | -1 | 96 | 98 | 104 | 86 | 44 |
| 13 | 80 | 81 | 65 | 83 | 86 | 8 |
| 14 | 69 | 68 | 67 | 83 | 71 | 7 |

TABLE III
RELATION BETWEEN NUMBER OF VMS AND $D_{std}$

| | $D_{std}(i)$ | | | | | |
|---|---|---|---|---|---|---|
| | Degree of Complexity | | | | | Mean($\sigma_{std}$)=24 |
| ($i$) VMs | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | $\sigma_{std}(i)$ |
| 5 | 58 | 69 | 94 | 73 | 80 | 49 |
| 6 | 48 | 37 | 79 | 63 | 61 | 38 |
| 7 | 42 | 43 | 39 | 71 | 48 | 31 |
| 8 | 78 | 23 | 60 | 34 | 40 | 30 |
| 9 | 46 | 9 | 41 | 44 | 32 | 21 |
| 10 | 32 | 23 | 39 | 20 | 34 | 18 |
| 11 | 42 | 25 | 31 | 24 | 26 | 18 |
| 12 | 41 | 26 | 26 | 28 | 24 | 17 |
| 13 | 19 | 32 | 15 | 26 | 22 | 13 |
| 14 | 0 | 37 | 15 | 24 | 20 | 11 |
| 14 | 21 | 18 | 22 | 11 | 22 | 11 |

the similar workload with over 9 VMs while avoiding choosing 11,12 VMs to satisfy the QoS requirement of application in real world.

## IV. SPARK IMPLEMENTATION AND EVALUATION : SCHEDULING JOBS BY ENTROPY GUIDED RESOURCE LOCAL ACTIVITY RANKING

Through the study from Section III, we understand the impact of complexity on the performance of cloud scheduling and how it lead to the violation of application's QoS requirements. We try to choose the suitable initial number of VMs to achieve more robust scheduling by understanding whether the system is in "Order" state or "Chaos" state. Generally speaking, complexity reduction is a way to improve QoS in cloud scheduling [19]. Although we can use simulation and try to reduce the complexity, however, there is limitation in this way since the simulation only models part of the complexity in the real world. In the real world cloud environment, there are complexity form of other media such as dynamic & unpredictable workload and heterogeneous links among the resources, which are hard to control or even uncontrollable during runtime. Relatively speaking, the cloud resources form of cloud is easier to control, as we can know its average performance from history by monitoring its CPU utilization. Learning from the concept of "taking human being as the essential to improve the quality of project management", we know the resource is the essential part to achieve better scheduling in the complex cloud. Thus, in this section, we will focus on resource-oriented complexity reduction.

### A. Spark Entropy Scheduler : New Approach To Better Satisfy QoS In Complex Cloud

Spark [21] is part of the Apache Software Foundation and claims speedups up to 100x faster than Hadoop MapReduce in-memory, or 10x faster on disk. The ability to bring response time of distributed data analysis into sub-second range has enabled powerful new application development - Cloud Analysis as a Service (CAaaS). In such case, user-facing services will be able to run sophisticated parallel computation, such as language translation, voice reorganization, highly search personalizations and context recommendation, on a per-query basis. However, when meeting with high concurrent of service query, the Spark performance become less reliable. Spark's performance is closely tied to its job scheduler. Most of the time, we need to deploy more resources to handling the increased service query, which will cause the increment of complexity in the scheduling system. Although the current scheduler in Spark works well in homogeneous environment with low query request, but it failed to better fulfil the QoS requirement of CAaaS as the cloud become more complex. If the scheduling strategy cannot provide an optimal way to guarantee the QoS, it will be difficult to popularize the service.

The current scheduler in Spark implicitly assumes that all the resource are homogeneous and local passive and randomly allocate resources to jobs. Without considering the local activity in cloud resource, such schedulers perform poorly when meeting the increasing complexity of the cloud.

In our proposed Entropy Scheduler, instead of randomly picking up resources, we first calculate the local activity ranking of all offered resources (Algorithm 2), and then schedule tasks inside a job according to the ranking. Tasks are scheduled with similar ranking resource so as to improve overall QoS satisfaction and reliability of scheduling performance.

**Algorithm 2** Calculate Resource Local Activity Ranking

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

### B. Experiments And Evaluation

In order to evaluate our proposed Entropy Scheduler, we conduct experiments on a private cloud with 3 heterogeneous

physical resource. The resource specifications and Spark configuration are shown on Table IV. A simple Spark application has been deployed on the server with the ability to accept user query to calculate $\pi$ with a predefined number of CPU cores concurrently. We use Apache Bench to load testing the Spark application under different schedulers (Our Entropy Scheduler and Spark Fair Scheduler [22]). The load testing will spawn a number of threads which continuously execute the same query. Each thread remains loaded and continues processing queries until all threads have finished, and the query response time of all requests from every thread will use for performance comparison.

TABLE IV
EXPERIMENTAL PLATFORM:RESOURCE SPECIFICATION

| Specification | Node 1 | Node 2 | Node 3 |
|---|---|---|---|
| Spark Role | Master&Worker | Worker | Worker |
| CPU | Xeon 3Ghz x 2 | Xeon 2.8Ghz x 2 | Xeon 1.8Ghz |
| Cores | 8 | 8 | 4 |
| RAM | 16GB | 12GB | 12GB |

*1) Experiment 1: Performance under Different Concurrent Level of HTTP Request Workload:* This experiment is used to verify the query response time and degree of satisfying of QoS requirement with Entropy Scheduler and Fair Scheduler under different concurrent level of request workload. The results are shown as follows in Figure 7, Figure 8 and Figure 9.

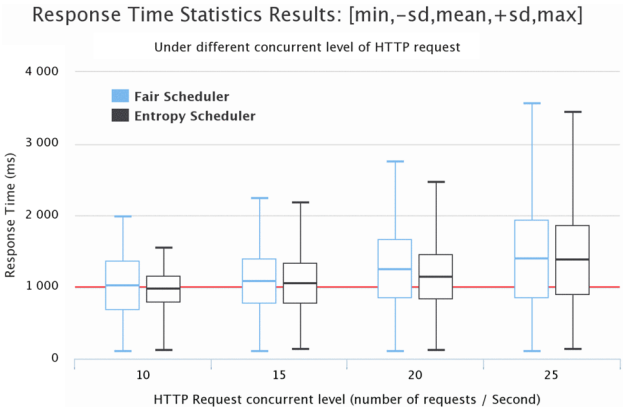


Fig. 7. Experiment 1: Response time statistics result

Figure 7 shows that Entropy Scheduler has better performance and a higher degree of satisfying of QoS requirement, which result in improvement of the overall server throughput as well (Figure 8).

However, increasing workload concurrency pose various challenges to the scheduling system. The cloud experience performance degradation with increasing workload concurrency. As seen from Figure 9, although Entropy Scheduler reduce a significant amount of failed requests compared with Fair Scheduler, it still has same performance bottlenecks inhibiting sub-second query response time which motivates future work of other optimization options.

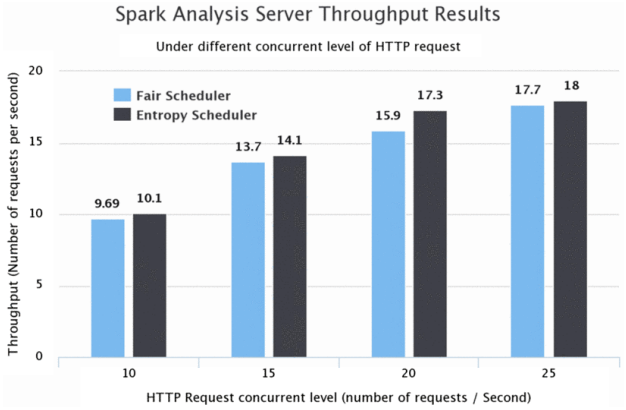


Fig. 8. Experiment 1: Spark analysis server throughput result

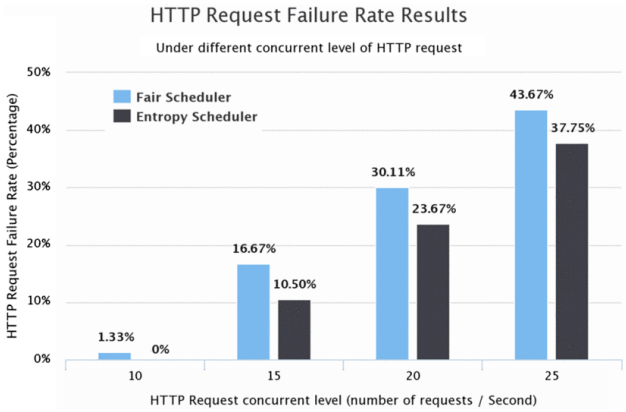


Fig. 9. Experiment 1: HTTP request failure rate result

*2) Experiment 2: Load Testing with 100,000 Query Requests at the Concurrent Level of 10:* Table V compare the various aspects of load testing result by each scheduler. Our results throughout the Evaluation section show Entropy Scheduler outperforms native Fair Scheduler in respect of QoS satisfaction. On average, in this heterogeneous cluster experiment, Entropy Scheduler is able to shorten the load testing completion time by 23%, reduce the average response time by 23% and standard deviation by 35%, and improve the overall server throughput by 30% compared with native Fair Scheduler.

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

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

Figure 10 indicates that 90% of queries are completed

within 1 second under Entropy Scheduler, while only 50% under Fair Scheduler. Such result shows that Entropy Scheduler is more capable of running CAaaS that providing web service with QoS guarantee.

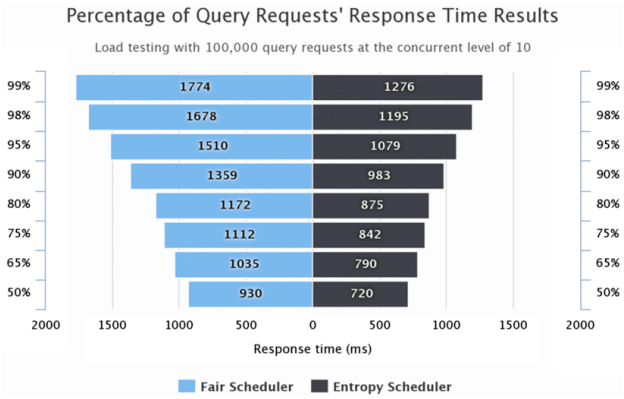


Fig. 10. Experiment 2: Percentage of the requests served within a certain time (Million Seconds)

### C. Discussion

Our experiments on 3 resources with 20 cores is small-scale, but the experimental results provide intuition for developing new scheduler based on entropy with large-scale of local active resources. From experiment 1, we have learned the critical bottleneck in current Spark Jobs Scheduling causing by handling high concurrent queries when the system complexity is increase. Compare with native Spark FAIR scheduler, Entropy Scheduler reduces the query Failure Rate by around 7%. The results in Experiment 2 show Entropy Scheduler out-perform FAIR Scheduler for CAaaS in complex cloud environment, which will be a starting point for future work, where we hope to run the low-latency query with better QoS guarantee.

## V. Conclusion And Future Work

The complexity is an important issue that affects QoS satisfaction bringing additional challenges to scheduling problem. In the present paper, the negative impact of complexity on deterministic cloud scheduling system was used to motivate the new scheduler development based on Entropy Theory to schedule tasks to resources involving local activity in the real world cloud. With the results in the 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.

Research on Complexity has just emerged in the area of cloud scheduling. The understandings of the origin of complexity (Locally-active cloud resource) and impact of complexity (Performance degradation, QoS guarantees violation and potential Chaotic behaviour) would offer useful information to find the limitation of current scheduling solutions and motivate new scheduler development under complex cloud environment. However, this paper focuses on the resource-oriented complexity. In the future, complexity raising from other media (etc. workload, links between resources, outer environment) are also need to be studied.

## References


[1] Chua, Leon O. "Local activity is the origin of complexity." International journal of bifurcation and chaos 15.11 (2005): 3435-3456.

[2] Bar-Yam, Yaneer. Dynamics of complex systems. Vol. 213. Reading, MA: Addison-Wesley, 1997.

[3] Boltzmann, Ludwig. "The second law of thermodynamics." Theoretical physics and philosophical problems. Springer Netherlands, 1974. 13-32.

[4] Plestys, Rimantas, et al. "The measurement of grid QoS parameters." Information Technology Interfaces, 2007. ITI 2007. 29th International Conference on. IEEE, 2007.

[5] Matthews, Robert AJ. "The science of Murphy's law." PROCEEDINGS-ROYAL INSTITUTION OF GREAT BRITAIN. Vol. 70. Oxford University Press, 1999.

[6] Zhang, Qi, Lu Cheng, and Raouf Boutaba. "Cloud computing: state-of-the-art and research challenges." Journal of internet services and applications 1.1 (2010): 7-18.

[7] Chen, Huankai, and Frank Z. Wang. "Spark on entropy: A reliable & efficient scheduler for low-latency parallel jobs in heterogeneous cloud." Local Computer Networks Conference Workshops (LCN Workshops), 2015 IEEE 40th. IEEE, 2015.

[8] Grassberger, Peter. "Damage spreading and critical exponents for model A Ising dynamics." Physica A: Statistical Mechanics and its Applications 214.4 (1995): 547-559.

[9] Bagnoli, F., R. Rechtman, and S. Ruffo. "Damage spreading and Lyapunov exponents in cellular automata." Physics Letters A 172.1 (1992): 34-38.

[10] Boccaletti, Stefano, et al. "The control of chaos: theory and applications." Physics reports 329.3 (2000): 103-197.

[11] Cambel, Ali Bulent. Applied chaos theory: A paradigm for complexity. Elsevier, 1992.

[12] Chua, Leon. Memristor, Hodgkin-Huxley, and edge of chaos. Springer International Publishing, 2014.

[13] Braun, Tracy D., et al. "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems." Journal of Parallel and Distributed computing 61.6 (2001): 810-837.

[14] Chen, Huankai, et al. "User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing." Parallel Computing Technologies (PARCOMPTECH), 2013 National Conference on. IEEE, 2013.

[15] Bala, Anju, and Inderveer Chana. "A survey of various workflow scheduling algorithms in cloud environment." 2nd National Conference on Information and Communication Technology (NCICT). 2011.

[16] Iosup, Alexandru, Nezih Yigitbasi, and Dick Epema. "On the performance variability of production cloud services." Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on. IEEE, 2011.

[17] Schad, Jrg, Jens Dittrich, and Jorge-Arnulfo Quian-Ruiz. "Runtime measurements in the cloud: observing, analyzing, and reducing variance." Proceedings of the VLDB Endowment 3.1-2 (2010): 460-471.

[18] Herroelen, Willy, and Roel Leus. "Project scheduling under uncertainty: Survey and research potentials." European journal of operational research 165.2 (2005): 289-306.

[19] Tndel, Petter, and Tor A. Johansen. "Complexity reduction in explicit linear model predictive control." Proc. of 15-th IFAC world congress. 2002.

[20] RRNYI, ALFRPED. "On measures of entropy and information." (1961).

[21] Zaharia, Matei, et al. "Spark: Cluster Computing with Working Sets." HotCloud 10 (2010): 10-10.

[22] Zaharia, Matei. "Job scheduling with the fair and capacity schedulers." Hadoop Summit 9 (2009).