International Journal of u- and e- Service, Science and Technology
Vol.8, No. 11 (2015), pp.415-424
http://dx.doi.org/10.14257/ijunesst.2015.8.11.39

Improve Constrained Recourse Fairness in Cloud Computing with Bottleneck-Aware Allocation

Jun Liu[1,a] and Xi Liu[2, b]

[1]*College of Mathematics and Information Science, Qujing Normal University, Qujing Yunnan China*
[2]*Colleg of Information Science and Engineering, Yunnan University, Kunming Yunnan China*
[a]*liujunxei@126.com,* [b] *lxghost@126.com*

Abstract

Fair resource is a key building block of any shared computing system. Recently fair division theory has emerged as a promising approach for the allocation of multiple computational resources among agents. Recent research has discussed efficiency and fairness requirements and identified a number if desirable scheduling objectives including so-called dominant resource fairness(DRF). DRF is not good between fair and resource utilization. A new allocation model Balancing Fairness and Efficiency with Bottleneck-Aware Allocation(BAA) find good appropriate balance between fairness to the clients and maximizing system utilization. But BAA without taking into account the weight value and maximum number of tasks which are users need to run. We propose the IBAA fair allocation mechanism. IBAA has lots of good properties, it satisfies DSI, PE and EF. We construct IBAA mechanisms that provably satisfy properties, and analyze the performance. We believe that our work informs the design of superior multiusers system, and at the same time expands the scope of fair division theory by initiating the study of dynamic and fair resource allocation mechanisms.

Keywords*: dominant resource fairness; cloud computing; resource allocation*

1. Introduction

Cloud computing[1]has become a hot research applications and is a new computing model which is the development of distributed computing, parallel computing and grid computing. The national institute of standards and technology to define the basic characteristics of cloud computing is on-demand self-service and rapid elasticity[2].To achieve these characteristics, the main use virtualization and its related technologies[3]. Modern datacenters are likely to be constructed from a variety of server classes, with different configurations in terms of processing capabilities, memory sizes, and storage spaces[4]. At any time, there are tens of thousands of clients concurrent running their high-performance computing applications on the shared computing system [5-9].

Resource allocation is a key building block of any shared computer system. One of the most popular allocation policies proposed so far has been max-min fairness [10], which maximizes the minimum allocation received by a user in the system. Assuming each user has an equal share of the resources. Dominant Resource Fairness(DRF) is a generalization of max-min fairness for multiple resources[11]. The intuition behind DRF is that in a multi-resource environment, the allocation of a user should be determined by the user's dominant share, which is the maximum share that the user has been allocated of any resources. For example, if user 1 runs CPU-heavy tasks and user 2 runs memory-heavy tasks, DRF attempts to equalize user 1's share of CPUs with user 2's share of memory. In the single resource case, DRF reduces to max-min fairness for that resource.

ISSN: 2005-4246 IJUNESST

A new model called Bottleneck Aware Allocation(BAA) based on the notion of local bottleneck sets to maximize system utilization while providing fairness in the allocations of the competing clients[12]. The allocations of BAA enjoy all of the fairness properties of DRF, like Sharing Incentive, Envy Freedom and Pareto Optimality. The model provides clients that are bottlenecked on the same device with allocations that are proportional to their fair shares, while allowing allocation ratios between clients in different bottleneck sets to be set by the allocator to maximize utilization.

2. Problem Definition

The storage system is composed of SSDs and HD arrays. SSDs and HDs are independent devices without frequent data migrations between them. We assume that we have two resources(CPU and Memory) in a cloud computing system.

2.1. Basic Setting

(1)The load of a client i on the CPU is h_i / c_s and on the Memory is m_i / c_d .

(2) Partition the clients into two sets D and S based on their hit ratios.

(3) Define the fair share of a client to be the throughput it gets if each of the resources are partitioned equally among all the clients. Denote the fair share of client i by f_i .

(4)Each user u_i requires r_{ij} -fraction of resource type j . For convenience, assume $r_{ij} > 0$ for all i , j . Let x_i be the number of tasks processed on the server for user u_i . Assume the number of tasks for user i need to be processed are B_i .

(5)Let A_i denote the allocation of client i under some resource partitioning. The total throughput of the system is $\sum_i A_i A_i = \sum_j r_{ij} B_i$.

(6)Assume that each user u_i has a publicly known weight w_i .

For now, we assume users have an finite number of tasks to be scheduled. Infinite users join system at different times.

2.2. Improve Constrained Resources Fairness Policy

BBA does not take into account the weight of the user, and whether the user has met the needs of resources to all the tasks to run. Defined as follows:

$$p_d = \frac{A_i}{f_i},\ p_s = \frac{A_j}{f_j},\ \frac{h_j}{h_i} \geq \frac{A_i}{A_j} \geq \frac{m_j}{m_i}.$$

(1)Fairness between clients in D

$\forall i, j \in D$, $\frac{\frac{A_i}{w_i}}{\frac{A_j}{w_j}} = \frac{f_i}{f_j}$, Define $p_d = \frac{A_i}{f_i w_i}$ to be the ratio of the allocation of client i to its fair share, $i \in D$.

(2)Fairness between clients in S

$\forall i, j \in S$, $\frac{\frac{A_i}{w_i}}{\frac{A_j}{w_j}} = \frac{f_i}{f_j}$, Define $p_s = \frac{A_j}{f_j w_j}$ to be the ratio of the allocation of client j to its fair share, $j \in D$.

(3)Fairness between a client in D and a client in S:

$$\forall i \in D,\ j \in S,\ \frac{h_j}{h_i} \geq \frac{\frac{A_i}{w_i}}{\frac{A_j}{w_j}} \geq \frac{m_j}{m_i}$$

We now formulate the objective function and constraints in terms of the auxiliary quantities p_d and p_s. $\forall i \in D$, $j \in S$.

$$\frac{h_j}{h_i} \geq \frac{p_d f_i w_i}{p_s f_j w_j} \geq \frac{m_j}{m_i} \Rightarrow \frac{h_j f_j w_j}{h_i f_i w_i} \geq \frac{p_d}{p_s} \geq \frac{m_j f_j w_j}{m_i f_i w_i} \Rightarrow \beta \geq \frac{p_d}{p_s} \geq \alpha .$$

Where $\alpha = \max_{i,j} \left\{ \frac{m_j f_j w_j}{m_i f_i w_i} \right\}$, $\beta = \min_{i,j} \left\{ \frac{h_j f_j w_j}{h_i f_i w_i} \right\}$.

We will give the optimization for allocation:
Maximize

$$\sum_{i \in D} A_i + \sum_{j \in S} A_j = p_d \sum_{i \in D} f_i w_i + p_s \sum_{j \in S} f_j w_j .$$

Subject to

$$A_i m_i + A_j m_j = p_d \sum_{i \in D} f_i w_i m_i + p_s \sum_{j \in S} f_j w_j m_j \leq c_d .$$

$$A_i f_i + A_j f_j = p_d \sum_{i \in D} f_i w_i h_i + p_s \sum_{j \in S} f_j w_j h_j \leq c_{ds} .$$

$$\beta \geq \frac{p_d}{p_s} \geq \alpha .$$

$$A_i = B_i(r_{i_1} \cdot r_{i_2}), \text{ if } B_i(r_{i_1} + r_{i_2}) < A_i .$$

3. Fairness Properties

The following are important and desirable properties of a fairness:

(1)Sharing incentive(SI). Each user should be better off sharing the cluster. Each user should not be able to allocate more tasks in a cluster partition consisting of $\frac{1}{n}$ of all resources.

(2) Dynamic Sharing incentive(DSI). For all users u_i: either $A_i = B_i(r_{i_1} + r_{i_2})$ or if $i, j \in D$ and $w_i \geq w_j$, then $A_i m_i \geq A_j m_j$. Else if $i, j \in S$ and $w_i \geq w_j$, then $A_i h_i \geq A_j h_j$.

(3)Envy-freeness(EF). For all users u_i : either $A_i = B_i(r_{i_1} + r_{i_2})$ or if $i, j \in D$, then $\frac{A_i m_i}{w_i} \geq \frac{A_j m_j}{w_j}$ or $i, j \in S$, then $\frac{A_i h_i}{w_i} \geq \frac{A_j h_j}{w_j}$. A client cannot increase its throughput by swapping its allocation with any other client. That is, clients prefer their own allocation over the allocation of any other client.

(4)Pareto efficiency(PE). It should not be possible to increase the allocation of a user without decreasing the allocation of at least another user. This property is important as it leads to maximizing system utilization subject to satisfying the other properties.

(5)Strategy-proofness (SP). Users should not be able to benefit by lying about their resource demands. This provides incentive compatibility, as a user cannot improve her allocation by lying.

Lemma 1. Let n be the number of clients. Then $f_i = \min\left\{\frac{c_d}{n \cdot m_i}, \frac{c_s}{n \cdot h_i}\right\}$. If $i \in D$, then $f_i = \frac{c_d}{n \cdot m_i}$, else if $i \in S$, the $f_i = \frac{c_s}{n \cdot h_i}$.

Proof. Identical to the proof of [BAA].

Lemma 2. All clients in a bottleneck set receive equal throughputs on the bottleneck device. Specifically, all clients in D receives $\frac{p_d c_d}{n} w_i$ from the Memory; and all clients in S receive $\frac{p_s c_s}{n} w_j$ from the CPU.

Proof. If $i \in D$, then $A_i m_i = p_d f_i w_i m_i = \frac{p_d c_d}{n} w_i$. Else if $j \in S$, then $A_i m_i = p_s f_j w_j m_j = \frac{p_s c_s}{n} w_j$.

Theorem 1. IBAA satisfies the DSI property

Proof. If $A_i = B_i(r_{i_1} + r_{i_2})$, user u_i get all of the resources to run all tasks. Otherwise, if $i, j \in D$, we have $A_i m_i = p_d f_i w_i m_i = \frac{p_d c_d}{n} w_i$, $A_j m_j = p_d f_j w_j m_j = \frac{p_d c_d}{n} w_j$. Because $w_i \geq w_j$, we have $A_i m_i \geq A_j m_j$. If $i, j \in S$, we have $A_i h_i \geq A_j h_j$ according to the same method of proof.

Theorem 2. IBAA satisfies PE property.

Proof. Identical to the proof of [BAA].

Theorem 3. IBAA satisfies EF property.

Proof. According to Lemma2, if $i, j \in D$, we have

$$\frac{c_d}{n m_i} m_i p_d \geq \frac{c_d}{n m_j} m_j p_d , \quad m_i p_d f_i \geq m_j p_d f_j , \quad \frac{m_i p_d w_i f_i}{w_i} \geq \frac{m_j p_d w_j f_j}{w_j} , \quad \frac{A_i m_i}{w_i} \geq \frac{A_j m_j}{w_j} .$$

If $i, j \in S$, we have $\frac{A_i h_i}{w_i} \geq \frac{A_j h_j}{w_j}$ according to the same method of proof.

Theorem 4. IBAA violates SP property.

Proof. For example, Consider a system with $c_d = 200$ IOPS, $c_s = 1000$ IOPS and four clients $p, q.r, s$ with hit ratios $h_p = 0.75$, $h_q = 0.5$, $h_r = 0.9$, $h_s = 0.95$. Assume $w_1 = w_2 = w_3 = w_4 = 1$. In this case, $h_{bal} = \frac{1000}{1200} \approx 0.83$. Hence, p and q are bottlenecked on the HD, while r and s are bottlenecked on the SSD: $D = \{p, q\}$ and $S = \{r, s\}$. So $f_p = 200$, $f_q = 100$, $f_r = 277.8$, $f_s = 263.2$. We have $\alpha = 0.55$ and $\beta = 1.67$. We have

$p_d = 1.41$, $p_s = 1.44$, $A_p = 282.5$, $A_q = 141.3$, $A_r = 398.6$, $A_s = 377.6$

according to (1). If user q lie about her ratio $h'_q = 0.6$. We have $\alpha = 0.55$ and

$\beta = 1.67$, $p_d = 1.44$, $p_s = 1.34$, $A'_q = 180$, $A_p = 289.4$, $A_r = 372.2$, $A_s = 352.7$

according to (2).

$$\begin{cases} Maximize \quad 300\ p_d + 541\ p_s \\ 100\ p_d + 41\ p_s \le 200 \\ 200\ p_d + 500\ p_s \le 1000 \\ 0.55 \le \dfrac{p_d}{p_s} \le 1.67 \end{cases} \tag{1}$$

$$\begin{cases} Maximize \quad 325\ p_d + 541\ p_s \\ 100\ p_d + 41\ p_s \le 200 \\ 225\ p_d + 500\ p_s \le 1000 \\ 0.55 \le \dfrac{p_d}{p_s} \le 1.67 \end{cases} \tag{2}$$

Algorithm1 shows the pseudo-code for IBBA scheduling algorithm. The output of this online algorithm are resources for each user according to compute p_d , p_s .

ALGORITHM1 IBBA pseudo-code

1: c_s : total CPU resources

2: c_d :total memory resources

3: $x = (x_1, x_2, \cdots, x_n)$ the number of tasks processed for user i .

4: $B = (B_1, B_2, \cdots, B_n)$ ：the number of tasks user need to run

5: $d = (d_{1r}, d_{2r}, \cdots, d_{nr})$: $r \in (CPU \ , Memory \ )$.

6: while wait new users or some users leave the system do

7: Maximize $p_d \sum_{i \in D} f_i + p_s \sum_{j \in S} f_j$

8: subject to

9: $p_d \sum_{i \in D} f_i m_i + p_s \sum_{j \in S} f_j m_j \le c_d$

10: $p_d \sum_{i \in D} f_i h_i + p_s \sum_{j \in S} f_j h_j \le c_d$

11: $x_i = \dfrac{p_d f_i h_i}{d_i} \le B_i$

12: $\beta \geq \frac{p_d}{p_s} \geq \alpha$

13: Foreach $u_i \in U$

14: if $u_i \in D$, then $A_i = p_d f_i$

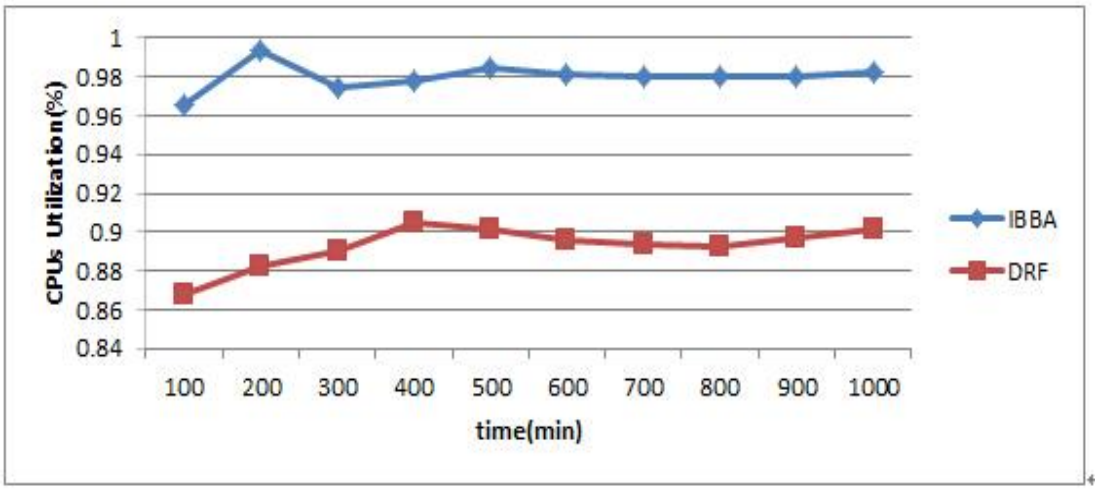
15: if $u_j \in S$, then $A_j = p_s f_j$

16: END

4. Experimental Results

In this section, we evaluate the performance of IBBA. For convenience, assume each user bring 1/n resources to the system. Every user adds to the system at different time and one or more users join the system at any point in time. Each user submits computing jobs, divided into a number of tasks, each requiring a set of resources. Assume $w_i = \frac{1}{n}$ for every user throughout this section. The requirement r_{ij} of each task are random numbers between 0 and 1, and B_i is a uniform random number in the range $[\frac{1}{nr_i}, \frac{1}{r_i}]$ for every user u_i , where $B_i \geq \frac{1}{nr_i}$ implies that user u_i needs to share the resources. Experimental platform environment is using C# in Visual studio 2010. The optimal solution is obtained by solving the linear program, where the software was written Lingo 10.

Figure 1, 2 and 3 show that random users added to the system with n=1000 users and m=2(CPU and Memory). Figure 1 shows IBBA and DRF of CPU and memory utilization. As can be seen from the Figure 1, the CPU utilization of IBBA are above DRF and memory utilization are below DRF. DRF often to ensure utilization of the CPU or memory of close to 100%, but cannot guarantee the utilization of other resources of close to 100%. The utilization of CPU and memory are all close to 100% in IBBA. Figure2 shows the ratio of the total number of tasks which user can run in DRF and IBBA. We can get from the figure 2, IBBA outperformed DRF. Figure3 shows the ratio of the minimum resource utilization. We have overall resource utilization of IBBA better than DRF according to Figure 3.



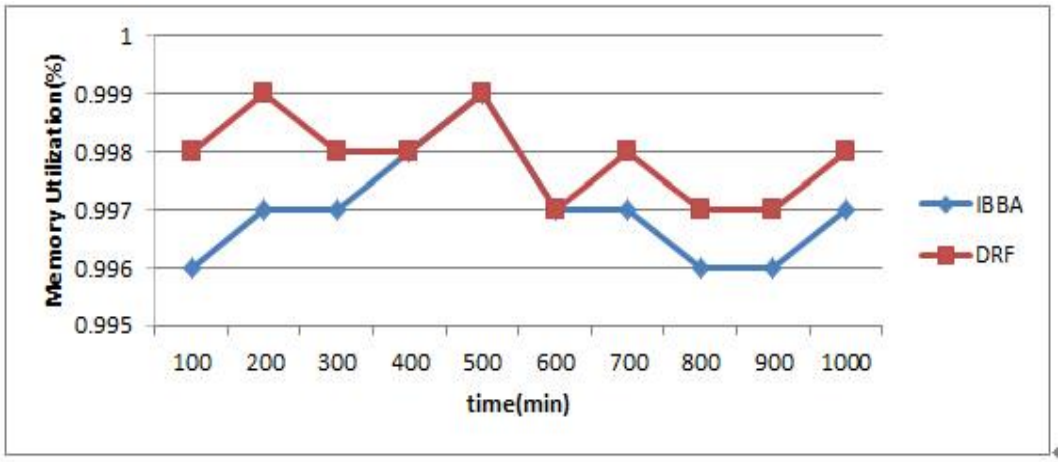


Figure 1. CPU and Memory Utilization

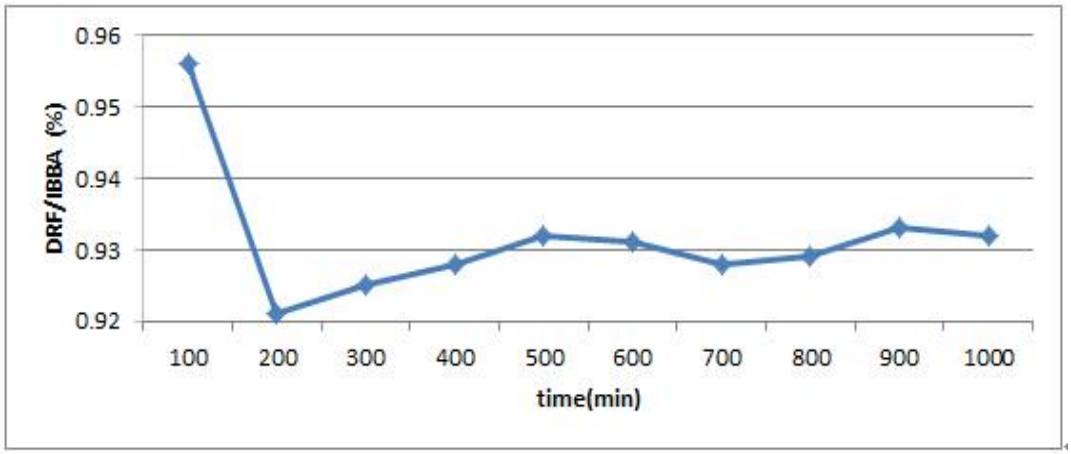


Figure 2. Ratio of Tasks Number

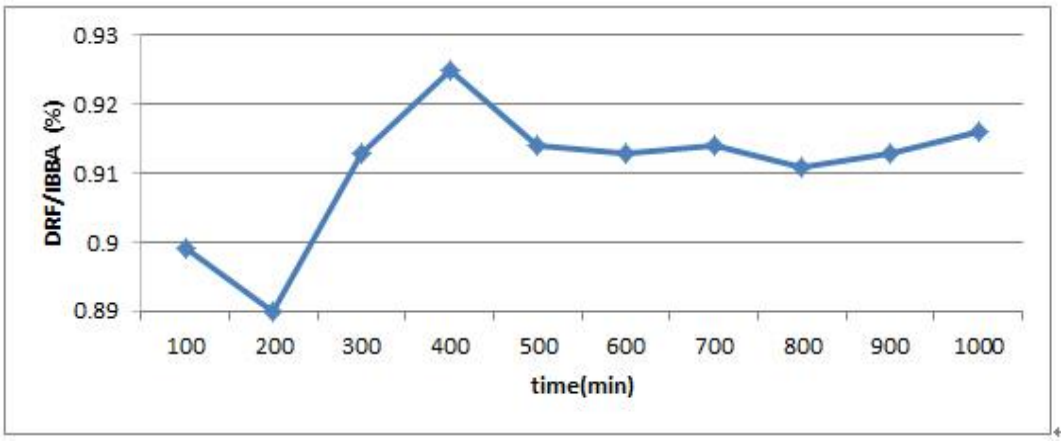


Figure 3. Ratio of the Minimum Resource Utilization

5. Conclusion and Future Work

In this paper, we study the problem which is users allowed different weight value with task number which is user need to run. We have introduced Improve Constrained Recourse Fairness with Bottleneck-Aware (IBBA), a fair sharing model that generalizes max-min fairness to multiple resource types. IBBA has lots of good properties, it satisfies

DSI , PE and EF. We give added weight to each user, so IBAA algorithm can distinguish considering the allocation of each user. And taking into account, in practice, The maximum number of tasks that her need to run is limited in practice, we take this into consideration and add the maximum number of tasks to each user who can get resources less than or equal to her maximum resources to run her all tasks avoiding the waste of resources.

BBA algorithm requires solving linear programming, but we have not a deterministic algorithm to solve. The system needs to re-solve when a user joins or leaves the system, so inevitably bring about increased migration and computation resources. We only take into account the CPU and memory resources, if the number of resources of more than two, linear programming style needs to be redefined. These are the next issue to be studied. As for future work, we use IBBA in the real system (e.g., Hadoop, yarn).

Acknowledgments

The work was supported by Chinese Natural Science Foundation Grant No.11361048.

References

[1] M. Ambrust, A. Fox and R. Griffith, " Above the Clouds: A Berkeley View of Cloud Computing[EB/OL]", (2011-01-25). http://www.eecs.berkeley.edu/pubs/techrpts/**2009**/ EECS-2009-28.pdf.
[2] J. T. Mell, National Institute of Standards and Techology, vol. 5, no. 20, **(2011)**.
[3] R. Buyya and S. C. Yeo, "Future Generation Computer System", vol. 2, no. 25, (**2009**).
[4] M. Armbust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, Commun. ACM, vol. 4, no. 53, (**2010**).
[5] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters. OSDI", vol. 4, **(2004)**.
[6] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker and I. Stoica, "Spark: Cluster Computing with Working Sets. HotCloud", vol. 10, **(2010)**.
[7] M. Isard, M. Budiu, Y. Yu, A. Birrell and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in Proceedings, EuroSys, **(2007)**.
[8] B. Hindman, A. Konwinski, M. Zahria, A. Ghodis, A. D. Joseph, R. Katz, S. Shenker and I. Stoica, "Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center, NSDI 2011", March, **(2011)**.
[9] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar and A. Goldberg, "Quincy: Fair Scheduling for Distributed Computing Clusters", I SOSP, vol. 09, **(2009)**.
[10] A. Ghodsi, M, Zaharia, S.Shenker and I.Stoica, "Choosy:Max-Min Fair Sharing for Datacenter Jobs with Constraints", EuroSys 2013, April **(2013)**.
[11] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker and I. Stoica, "Dominant resource fairness: fair allocation of multiple resource types", Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation, NSDI'11, vol. 8, **(2011)**.
[12] H. Wang and P. J. Varman, "Balancing Fairness and Efficiency in Tiered Storage System with Bottleneck-Aware Allocation", Proceedings of the USENIX Conference on File and Storage Technologies(FAST), Feb, **(2014)** , pp. 229-242.

Authors

Jun Liu
Professor
College of Mathematics and Information Science, Qujing Normal University, Qujing Yunnan China

Xi Liu
Doctor
College of Information Science and Engineering, Yunnan University, Kunming Yunnan China