Multi-Level Queue Dominant Resource Fairness in Cloud Computing

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

[1] College of Mathematics and Information Science, Qujing Normal University,

Qujing Yunnan China 655011

[2]College of Information Science and Engineering, Yunnan University,

Kunming Yunnan China 650091

[a]liiujunxei@126.com, [b]lxghost@126.com

Keywords: Dominant resource fairness; Multi-level queue; Cloud computing.

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 users. While in reality users are not all present in the system simultaneously. Dominant Resource Fairness(DRF) satisfies SI , PE AND EF, but violates SP. We have introduced Multi-level Queue Dominant Resource Fairness (MQDRF), a fair sharing model that generalizes max-min fairness to multiple resource types. MQDRF has lots of good properties, it satisfies DSI, PE, EF and SP. We construct MQDRF mechanisms that provably satisfy properties, and analyze the performance.

Introduction

Cloud computing[9] 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[10].To achieve these characteristics, the main use virtualization and its related technologies[11]. Cloud computing has emerged as a popular computing paradigm that delivers resources to users. Resource utilization is a key design issue in the cloud for both users and cloud providers. Current supercomputers and data centers typically consist of thousands of servers connected via a high-speed network. 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(e.g., MapReduce[2], MPI, Spark[3], Dryad[5],Mesos[7],Choosy[1],Quincy[8]) on the shared computing system.

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[1], 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)[6] is a generalization of max-min fairness for multiple resources. 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.

Nevertheless, some aspects of realistic computing system are beyond the current scope of fair division theory. Perhaps most importantly, the literature does not capture the dynamics of these systems. Indeed, it is typically not the case that all the users are present in the system at any given time; users may arrive and depart, and the system must be able to adjust the allocation of resources. For example, if one user arrives before another, the first user should intuitively have priority. What does fairness mean in this concept? We introduce the concepts that are necessary to answer this question, and design the mechanisms this satisfy our proposed desiderata.

Problem Definition

2.1 Basic Setting

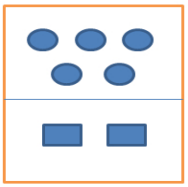
In a cloud computing system, let $c = (c_1, c_2, \ldots, c_m)$ be its resource capacity vector. Where each element c_r denotes the total amount of resource r available in server. Without loss of generality, for every resource r, we normalize the total capacity of server to 1.

$c = (c_1, c_2, \ldots, c_m) = (1,1, \ldots, 1)$

Let U={1,…,N} be the set of cloud users. For every user i, we normalize the resource demand vector to d_i, where d_{ir} is the fraction of resource r required by each task of user i over the entire system. For simplicity, we assume positive demands for all users, $d_{ir} > 0, \forall i \in U, r \in R$. Let x_i be the number of tasks processed on the server for user i. User i joins the system at time t_i^s. Let A_i be the allocation returned when user i reports its resource demand d_i and A_i^t be the allocation returned from the time t. The resource requirement constraints are

$$\sum_{i=1}^{n} d_{ir} x_i \le c_r, \quad r = 1,2,\cdots,m \tag{1}$$

$$D_i = \max\left\{\frac{d_{ir} x_i}{c_r}, r = 1,2,\cdots,m\right\}, \quad r = 1,2,\cdots,n \tag{2}$$



Server
(2 CPUs,5GB)

Fig. 1. An example of a system containing a server shared by two users

As a concrete example, consider Fig.1. The system contains 2CPUs and 5GB memory, the normalized capacity vectors of server is $c=(c_1,c_2)=(1,1)$. Now suppose there are two users. User 1 has memory-intensive tasks each requiring 0.2 CPU time and 1GB memory, while user 2 has CPU-heavy tasks each requiring 1 GB time and 0.2GB memory. The demand vector of user 1 is $d_1=(d_{11},d_{12})=(0.2/2,1/5)=(0.1,0.2)$. User 2 has $d_2=(d_{21},d_{22})=(1/2,0.2/5)=(0.5,0.04)$.

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

2.2 Multi-level Queue Dominant Resource Fairness Definition

For every user, DRF computes the share resources allocated to that user. The maximum among all shares of a user is called that user's dominant share, and the resource corresponding to the dominant share is called the dominant resources. DRF simply applies max-min fairness across uses' dominant shares. That is, DRF seeks to maximize the smallest dominant share in the system, then the second-smallest, and so on.

Definition 1. Multi-level Queue Dominant Resource Fairness (MDRDF) is a multi-level queue model. MDRDF has K levels and each level has DK_k, u_k and N_k. u_k is a set of users and N_k is maximum number of users in level k. New user is add to the queue level 1 and the users in queue level k-1 are placed to queue level k(k=1,2,…,K) at every time t. We use DRF allocation policy at each level queue. MDRDF satisfy the following constraints:

$$Dk_1 \ge Dk_2 \ge \cdots \ge Dk_k . \tag{3}$$

$$N_1 = N_2 = \cdots = N_{k-1} \le N_k . \tag{4}$$

$$\sum_{i\in u_k}\frac{d_{ir}x_i}{c_r}\le\frac{Dk_k}{N_k},\quad k=1,2,\cdots,k,\quad r=1,2,\cdots,m. \tag{5}$$

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). New user is add to the queue level 1 and the users in queue level k-1 are placed to queue level k(k=1,2,…,K) at every time t. When the queues are full, MQDRF satisfies SI in each level queue.

(3)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.

(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)Envy-freeness(EF). A user should not prefer the allocation of another user. If $t_i^s \ge t_j^s\ \forall i,j\in U$, then $A_i \ge A_j^{t_i^s}$. If $t_i^s < t_j^s\ \forall i,j\in U$, then $A_i \ge A_j$.

Consider the MQDRF vector $\mathrm{D}=(DK_1,DK_2,\ldots,DK_K), N=(N_1,N_2,\ldots,N_K)$. Next we will prove MQDRF satisfies DSI.

Theorem 1. MQRDF satisfies the DSI property.

Proof. When the queues are full. $\forall i,j\in U,\ K_i=K_j,$ we have

$$\frac{DK_{k_i}}{N_{k_i}}=\frac{DK_{k_j}}{N_{k_j}}=\max\frac{d_{ir}x_i}{c_r}=\max\frac{d_{jr}x_j}{c_r},\quad r=1,2,\cdots,m.$$

The amount of the resource in the queue level k_i is $D_{k_i}N_{k_i}$, the radio between the amount of resource r user i can use in the queue level k_i and the total of that resource available in the queue level k_i is:

$$\frac{d_{ir}x_i}{c_r^{k_r}}=\frac{Dk_{k_i}c_r}{Dk_{k_i}N_{k_i}}=\frac{Dk_{k_i}c_r}{Dk_{k_i}\left|U_{k_i}\right|}=\frac{1}{\left|U_{k_i}\right|}.$$

r is a dominant resource of user i. For user j, we have the above formulas.

Allocation policy is DRF at each level queue and DRF satisfies SI, so MQDRF satisfies SI in each level queue.

Theorem 2. MQRDF satisfies SP property.

Proof. If there is a user i, let x_i be the number of tasks processed when user I truthfully reports its resource vector d_i, and let x_i' be the number of tasks processed when user i misreports by $d_i'\ne d_i$. Combining (2) and (5), we have

$$\frac{Dk_{k_i}'}{N_{k_i}}=\max\frac{d_{ir}'x_i'}{c_r}=\frac{Dk_{k_r}}{N_{k_r}}=\max\frac{d_{ir}x_i}{c_r},\quad \text{if}\quad d_i'\ge d_i.$$

then $x_i'\le x_i$, else $x_i'>x_i$. User i cannot increase its resources by misreports. For arbitrary honest user $\forall j\in U$, we have $d_{jr}x_j'=Dk_{k_j}c_j/N_{k_i}=d_{jr}x_j$.

Theorem 3. MQRDF satisfies PE property.

Proof. Assuming MDRDF does not satisfy PE property, we must have

$$\forall i \in U,\ \exists \varepsilon > 0,\ \sum_{g=1}^{n} d_{gr} x_g + \varepsilon \le 1.$$

which implies that

$$x'_i = \frac{d_i x_i + \varepsilon}{d_i}, \mathrm{D}' = \frac{d_i x'_i}{c} > \frac{d_i x_i}{c} = DK_{K_i}/N_{K_i}.$$

it will contradict the fact that D is the MQRDF vector and violates that DRF satisfies PE property.

Theorem 4. MQRDF satisfies EF property.

Proof. if $t_i^s \ge t_j^s$, $\forall i, j \in U$, we must have $k_i \le k_j$. $A_i = \sum_{k=K_i}^{K-1} tDK_k/N_k + \sum_{k=1} tDK_K/N_K$.

$$A_j^{t_i^s} = \sum_{k=k_j}^{k-1} t\,Dk_k/N_k + (k_j - k_i)tDk_k/N_k + \sum_{k=1} t\,Dk_k/N_k. \quad DK_k \ge DK_K, k = 1,2\ldots,K,$$

according to (3).

$$A_i - A_j^{t_i^s} = \sum_{k=k_i}^{k-1} tDk_k/N_k + \sum_{k=1} tDk_k/N_k - (\sum_{k=k_j}^{k-1} tDk_k/N_k + (k_j - k_i)tDk_k/N_k + \sum_{k=1} tDk_k/N_k)$$

$$= \sum_{k=k_i}^{k_j-1} tDk_k/N_k - \sum_{k=k_i}^{k_j-1} tDk_k/N_k \ge 0 \qquad (6)$$

if $t_i^s < t_j^s\ \forall i, j \in U$, we must have $k_i > k_j$. we have

$$A_i - A_j = \sum_{k=1}^{k-1} tDk_k/N_k + \sum_{k=1} tDk_k/N_k + (k_i - k_j)tDk_k/N_k - (\sum_{k=1}^{k-1} tDk_k/N_k + \sum_{k=1} tDk_k/N_k)$$

$$= (k_i - k_j)tDk_k/N_k.$$

Experimental Results

In this section, we evaluate the performance of MQDRF. The CPUs and memory of server are normalized so that the maximum server is 1. Each user submits computing jobs, divided into a number of tasks, each requiring a set of resources.

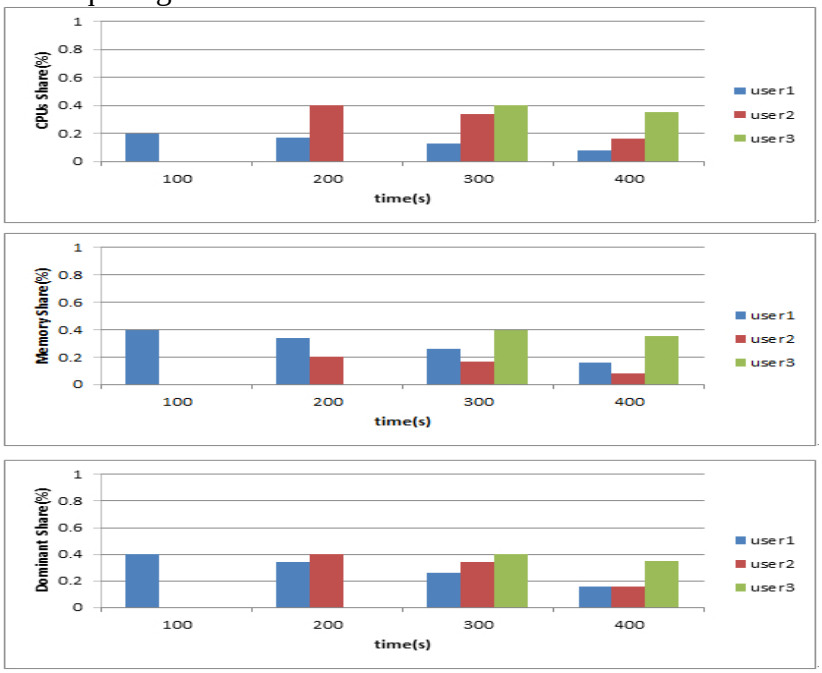


Fig. 2. CPU, memory, and global dominant share for three users

Dynamic allocation: Our first evaluation focus on the allocation fairness of the proposed MQDRF when users dynamically join the system. We simulate 3 users join the system at different time with different resource requirements. User 1 joins the system at the begging, requiring 0.01 CPU and 0.02 memory for each of its task. As shown in Fig.2, since only user 1 is active at the beginning, it is allocated 20% CPU and 40% memory. This allocation continue until 100s, at which time user 2 joins and submits CPU-heavy tasks, each requiring 0.02 CPU and 0.01 memory. So user 1 is placed to queue 2 and user 2 adds to queue 1. At 300s, user 3 joins the system and submits tasks, each requiring 0.01 CPU and 0.01 memory. User 3 adds to queue 1 , user 2 is placed to queue 2 and user 3 is placed to queue 3.

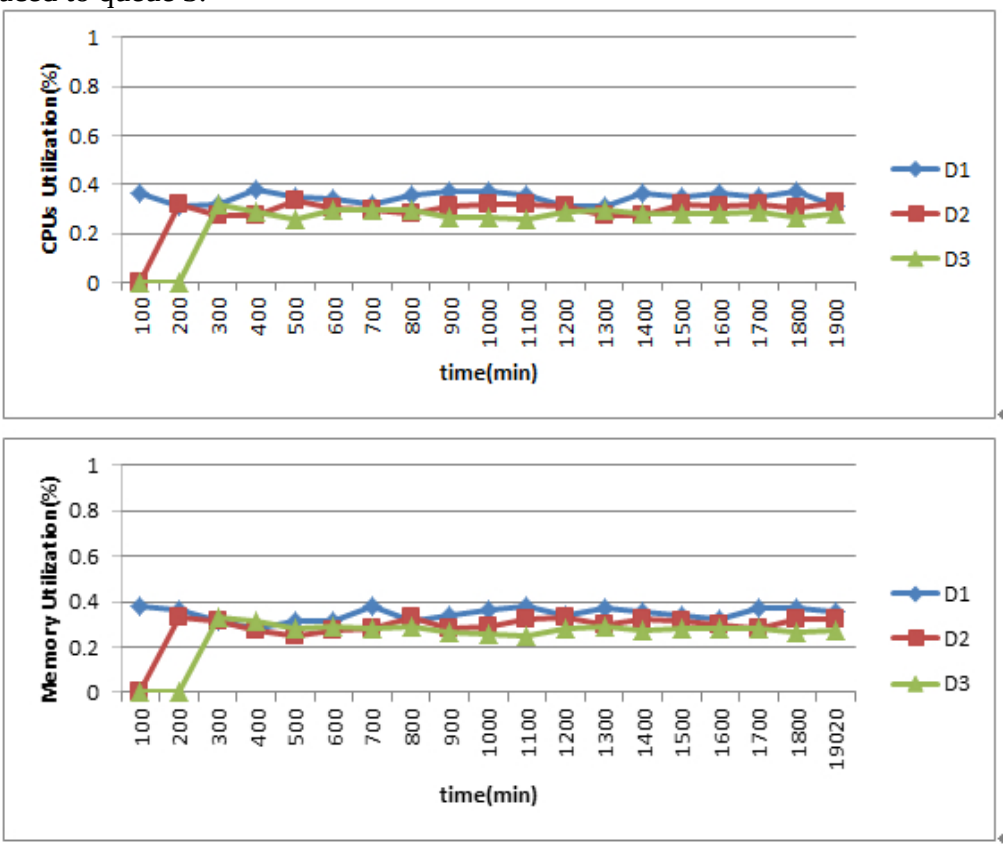


Fig .3. CPU and memory share for three queues

Resource utilization: We next evaluate the resource utilization of the proposed MQDRF algorithm. We take infinite users to join the system at different times. Fig.5 depicts the time series of CPU and memory utilization of the three queues.

Conclusion and Future Work

In this paper, we study the problem which is users arrive at the different times. We have introduced Multi-level Queue Dominant Resource Fairness (MQDRF), a fair sharing model that generalizes max-min fairness to multiple resource types. DRF satisfies SI , PE AND EF, but violates SP. MQDRF has lots of good properties, it satisfies DSI , PE, EF and SP. While in reality users are not all present in the system simultaneously, users which arrive at system at different times are allocated resources unfair. Any user is allocated fair resources in MQDRF.

As for future work, we use MQDRF in the real system (e.g., Hadoop, yarn) and Dynamic adjustment Multi-level Queue.

Acknowledgements

The work was supported by Chinese Natural Science Foundation Grant No.11361048, Qujing Normal University Natural Science Foundation Grant No.2012QN016.

References

[1] A.Ghodsi, M, Zaharia, S.Shenker and I.Stoica. EuroSys Vol. 23, (2013),p.78

[2] J.Dean and S.Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. OSDI'04, (2004)

[3] M. Zaharia, M. Chowdhury, M.J.Franklin, S.Shenker, I.Stoica. Spark: Cluster Computing with Working Sets. HotCloud Vol.10,(2010),p.10

[4] M. Armbust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," Commun. ACM,Vol.53, (2010), p. 50

[5] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D.Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in Proc. EuroSys,(2007)

[6] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," in Proc. USENIX NSDL,(2011)

[7] 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)

[8] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, A. Goldberg. Quincy: Fair Scheduling for Distributed Computing Clusters, In SOSP, Vol.9, (2009),p.261

[9] M.Ambrust, A.Fox, 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)

[10] J.Mell,T.Grance. The NIST definition of cloud computing. [S.I.]:National Institute of Standards and Techology,(2011)

[11] R.Buyya,C.S.Yeo. Future Generation Computer System, Vol.25,(2009), p:599