Multi Attribute D-S Evidence Theory Based OCC for Shared-State Scheduling in Large Scale Cluster

https://doi.org/10.3991/ijoe.v12i12.6457

Libo He, Zhenping Qiang Wei Zhou, and Shaowen Yao
Yunnan University, Kunming City, China

***Abstract*—With the growth of big data problems, nowadays the size of cloud-scale computing clusters is growing rapidly to run complicated parallel processing jobs. To full utilize cluster resources, the cluster management system is being challenged by the scaling cloud size and the often more complicated application requirements. Omega scheduling software provides a flexible and scalable shared-state scheduling architecture for large scale cluster scheduling. One of its key ideas is using an optimistic concurrency control (OCC) algorithm to let parallel schedulers concurrently make decisions. However, there are few studies exploring to extend OCC for a shared-state scheduling architecture. Furthermore, most of the traditional' shared-state scheduling architectures also use the same OCCs as Omega does. In this paper, we present a multi attribute Dempster–Shafer (D-S) evidence theory based OCC for shared-state scheduling. This OCC adapts the multi attribute D-S evidence theory to help making conflict decisions for some scheduling transactions. Experiments' results show that our method can obtain in some respects more optimized scheduling results compared to coarse-grained conflict detection of Omega.**

***Index Terms*—large scale cluster scheduling; multi attribute D-S evidence theory; optimistic concurrency control; Shared-state scheduling**

I. INTRODUCTION

With the coming of the era "Big Data", nowadays tens of thousands of machines are integrated into cloud infrastructure to provide different services. However, the scaling cluster size and more complicated application requirements let efficient scheduling to be a challenge for resource management system. Therefore, many schedulers, such as Hadoop Scheduler [1], Mesos [2], YARN [3], Omega [4], Sparrow [5] and Apollo [6], Borg [7], Torcil [8], Hawk [9] have been emerged in recent years.

Omega, which is the first to propose the notion of shared-state and introduces OCC into scheduling architecture, is a successful implementation and its notion has been used in the fellow shared-state schedulers. Yet there is no centralized control in shared-state scheduling architecture, so global optimization is not workable in this architecture. Priority preemptive mechanism is used by some shared-state scheduling to ensure fairness between jobs. However, there is few studies aim at how to optimize same priority band jobs to achieve better performance.

Rather than using OCC to maintain serial equivalence, we explore to use it to achieve some optimized goals. Omega adopts two OCC methods: coarse-grained conflict detection and fine-grained conflict detection. However, for this case, both of them may not effective. So we propose a new OCC method to deal with this case.

The contributions of this paper are as follows:

1) We propose an OCC method, which based on multiple attribute D-S evidence theory and OCC-Sacrifice mechanisms, for shared-state scheduling architecture. To the best of our knowledge, it is the first work to propose and adopt this method in shared- state scheduling.

2) To achieve some optimized goals, we set up a multi attribute model and use D-S theory to combine those attribute to help OCC make conflict decision.

3) We rewrite the Omega's simulator and implement this OCC in the simulator. And then we design a series of experiments to evaluate it.

The rest of this paper is organized as fellows. Section II presents background of our research. Section III describes the mechanisms of our OCC method. We present and analyze experiments results in section IV. Section V describes our future work.

II. BACKGROUND AND MOTIVATION

Scheduling in cluster has been extensively explored before. However, with the continuous scaling of cluster size, monolithic schedulers such as Hadoop Scheduler cannot meet the requirement of scalability. Two-level schedulers, such as Yarn and Mesos, distributed schedulers, such as Omega[4], Sparrow[5] and Apollo[6], Borg[7], Torcil[8], and hybrid schedulers, such as Hawk[9], are proposed to figure out some solutions to this problem. Due to decentralized decisions, the distributed schedulers are more favorable for efficient scheduling in large scale clusters. Therefore, researchers and developers pay more attentions to distributed schedulers recently.

Omega is a flexible and effective scheduling solution for large scale computing cluster. Shared-state and OCC are two key notions in Omega. Some fellow schedulers also use the same notions as Omega. The purpose of using OCC in shared-state scheduling is to maintain integrity of cell state and serial equivalence [10, 11] of concurrent schedule transactions. Since Kung and Robinson firstly proposed Optimistic concurrency control in 1981 [12]. So far, many OCC algorithms have been proposed for centralized database system and distributed database system [13, 14, 15]. Some OCC algorithms, such as OCC-Sacrifice [16], OCC-Broadcast Commit [17], MVCC

(Multi Version Concurrency Control) [18] have been widely used in many fields.

The OCCs which are adopted in Omega is fine-grained conflict detection and coarse-grained conflict detection. Coarse-grained conflict detection is similar to backward validation and more suitable for users who have stricter requirement about serial equivalence [10, 11]. Fine-grained conflict detection is a completed relaxed method which has no concurrency control mechanism.

When we use shared-state scheduling architecture and these OCCs in practical application, we find some optimized goals are hard to be realized. For example, we want to reduce the number of abandoned jobs which only have a little fraction of tasks have not been successful scheduled. Preemptive scheduling support higher priority tasks to preempt the resource of lower priority task even the scheduling transaction of higher priority tasks have been validated as conflict by OCCs. In Google's current resource management system, to eliminate preemption cascades, production priority band tasks are disallowed to preempt one another. Besides of preemptive scheduling, we try to explore to find a solution to do some optimized operation for the same priority band tasks.

The notion of OCC-Sacrifice inspires us. When two concurrent scheduling transactions have been validated in the same time, OCC-Sacrifice can let the less important transaction to be conflict. Then, we can do optimize operation for that part of scheduling transactions if we can judge which transaction is important for our optimized goals. However, due to the unpredictability of scheduling decision, accurately judging the importance of scheduling transactions in the same priority band is a challenge.

Since weight coefficients can be learned from workload trace by learning algorithm. How to set the combination rule for those weights is need carefully consideration. From the experiences of previous research, compare to simple linear weighted combination method, D-S evidence theory is more efficient method to improve accuracy of the weight [19, 20, 21]. So in this paper, we explore to combine D-S evidence theory [22, 23, 21] and OCC-Sacrifice to let our OCC achieved partly optimized goals in some respects.

We will explain the detail mechanism of our method in section III.

III. MULTIPLE ATTRIBUTE D-S FUSION BASED OCC

A. Multiple attribute fusion based on D-S evidence theory

We suppose tasks have three attributes that influence the importance of tasks. Then we assume the values of those attributes are p_1, p_2, p_3, which are normalized to satisfy $-1 \le p \le 1$. Furthermore, we assume the weight coefficients degrees of those attributes are η_1, η_2, η_3, and let $-1 \le \eta \le 1$. We define $\mu_i = \eta_i * p_i (i = 1,2,3)$ and combine μ_i by a kind of evidence combination rule which is presented in formula (1).

When we use the formula 1, we should firstly judge if those μ satisfy the following combination principles. It is obvious that they can meet the first three principles. If two attribute positively correlate to final attribute, their combinative value would be bigger than either of them. This result satisfies the principle 4. Otherwise if two attribute negatively correlate to final attribute, their combinative final attribute value will equal or smaller than both of them. The final value also meets the principle 5. If the μ_i is -1 or 1, then the final value would also be 1. Then the principle 6 could be also satisfied.

1. $\mu_1 \oplus \mu_2 = \mu_2 \oplus \mu_1$
2. $(\mu_1 \oplus \mu_2) \oplus \mu_3 = \mu_1 \oplus (\mu_2 \oplus \mu_3)$
3. $|\mu_1 \oplus \mu_2| \le 1$
4. $\mu_1 \oplus \mu_2 \ge \max(\mu_1, \mu_2), \mu_1 \ge 0, \mu_2 \ge 0$
5. $\mu_1 \oplus \mu_2 \le \min(\mu_1, \mu_2), \mu_1 \le 0, \mu_2 \le 0$
6. $\mu_1 \oplus \mu_2 = 1, |\mu_1| = 1$

$$(\mu_1) \oplus (\mu_2) = \mu_1 + \mu_2 + \mathrm{sig}(\mu_1 * \mu_2)$$

$$\mathrm{sig} = \begin{cases} -1, (\mu_1 > 0 \wedge \mu_2 > 0) \vee (\mu_1 > 0 \wedge \mu_2 < 0 \wedge |\mu_1| > |\mu_2|) \\ 1, (\mu_1 < 0 \wedge \mu_2 < 0) \vee (\mu_1 > 0 \wedge \mu_2 < 0 \wedge |\mu_2| > |\mu_1|) \\ 0, (\mu_1 < 0 \wedge \mu_2 > 0 \wedge |\mu_1| = |\mu_2|) \end{cases} \quad (1)$$

According to different attribute, the normalized methods of their value are different. In our simulation, the requested resource and runtime of tasks are known in advance. So, we preliminary consider three types of task attribute in our experiments. The task attribute p_1 is scheduled completion related. It can be calculated by the number of successfully scheduled tasks in a job divided by the total number of tasks in a job. The task attribute p_2 is requested resource related and the task attribute p_3 is task runtime related. Both of them can be calculated from the statistic of historical data. For p_2, we assume the max resource (CPU) of the machine is r_{max} and r_{cur} is the current requested resource of a task. Then p_2 is normalized by formula (2)

$$p_2 = \frac{r_{max} - r_{cur}}{r_{max}} \quad (2)$$

For p_3, we assume historical max duration value of tasks in a job is $d_{max}$and min value is d_{min}. Then p_3 can be normalized by formula 3.

$$p_3 = \frac{d_{cur} - d_{min}}{d_{max} - d_{min}} \quad (3)$$

The parameter η depends on the final goal of scheduling. According to different application requirement, different value can be given.

B. Multiple attribute D-S fusion based OCC

In order to implement our method in Omega's simulator, except cell state, another global data object which called placement array is used to record concurrent preliminary placement claims information and validate conflict. The maintenance of placement array is like cell state except the update operation. There is no concurrent update operation for a claim in placement array, so it isn't need concurrency control method.

As showed in TABLE I, SC is shared cell state. PA is placement array and C_{claims} is task-machine placement claims which created by schedulers through scheduling algorithm. Those claims are added to placement array after the end of schedule phase.

TABLE I.
ALGORITHM 1

Input:	**SC – shared cell state** **PA – placement array** **C_{claims}– task-machine claims created by scheduling algorithm**
1	**foreach** C_i in C_{claims}
2	C_i.importance = combine(p_1,p_2, p_3,η_1, η_2, η_3)
3	**def** condition1 = (C_i.sn != C_k.sn && C_i.ms ==C_k.ms &&C_i.bv == true)
4	**def** condition2 = (C_i.sn != C_k.sn && C_i.ms ==C_k.ms &&C_i.bv == false && C_i. importance < C_k. importance)
5	**if** C_i.ms != SC.ms **then**
6	$C_{conflict}$.add (C_i)
7	SA.delete (C_i)
8	**else if** SA[m].filter(C_i (condition1|| condition2)).size >0 **then**
9	$C_{conflict}$.add (C_i)
10	SA.delete (C_i)
11	**else then**
12	$C_{nonconflict}$.add (C_i)
13	SC.assignResources(C_i)
14	**def** $fc_{obsolete}$ = (C_k.tid !=C_i.tid)&&(C_k.ms<=C_i.ms)
15	$C_{obsolete}$ =SA.filter(C_i,$fc_{obsolete}$)
16	SA.delete($C_{obsolete}$)
17	**end if**
18	**end foreach**

In shared-state scheduling, task-machine placement claims are committed by schedulers in the form of transactions. Whether those transactions succeed or not is decided by OCC which is Algorithm1 in this paper. Algorithm1 separately validates claims. For each claim C_i, it firstly uses D-S evidence theory to combine multi attribute value to one importance factor $C_{i\text{-}importance}$ (line 1-2). Then define two conflict validate conditions (line 3-4). The condition2 is used to let lower important transaction to be conflict when more than two claims request the same resource concurrently. By those validate conditions, claims are judged as conflict or non-conflict (line 6-18). For conflict claims, they are added to conflict set and their related information is deleted from place array. On the other side, for non-conflict claims, they are added to non-conflict set. Their related tasks are assigned required resource and their obsoleted information is deleted from place array.

IV. EXPERIMENTS EVALUATION

In this section, in order to evaluate the effectiveness of our method, Omega's lightweight simulator which is written in Scale is been extended to implement our method. The runtime environment of our simulation experiments is JVM (openjdk-6-jdk) and Scala 2.9.0 based on a 64 bit Ubuntu 10.04 Server. The hardware environment is setup on one Intel(R) Core(TM) i7-3612QM CPU with total 4 cores, and 6GB of RAM.

Previous papers [24, 25 and 26] describe and analyze some computing cluster's trace. In our experiments, we use the Google production trace to synthesize a production priority band workload by considering some job's characteristics: the mean per-task resource, mean task duration, the mean number of tasks per job, and job inter-arrival time.

In order to simulate scheduling jobs in a cell, we construct a cell environment as following: 10000 machines, each machine have 4 CPU cores and 16 GB memory.

We use three metrics to evaluate our method:

1) Successful transactions fraction

In the paper, in order to find the influence of different η, we define successful transactions fraction as the number of successful transaction divided by the total number of transaction in a special value of p_1. For example, every task has a scheduled completion related attribute p_1 which is defined as the number of successful scheduled task divided by the total number of tasks in a job. When $p_1 = 0$, the successful transactions fraction which relate to p_1 is the number of successful task transactions whose value of p_1 is zero divided by the total number of transaction whose value of p_1 is zero.

2) Number of abandoned jobs

In application, it may have some unfortunate jobs which will be never scheduled completely after a large number of scheduling attempts due to conflicts. So in the simulator, we limit any single job to 1,000 scheduling attempts, and abandon job is the job which has not been completely scheduled after 1000 tries or which has not successfully been scheduled one task in 100 tries. Abandoned job number is the total number of abandoned jobs.

3) Job queuing time

In this paper, we define the job queuing time as the average time jobs takes to queue in pending queue till jobs have been fully scheduled.

In our OCC method, a claim is validated as conflict under the follow three cases:

Case1: the sequence number of the machine which is required by the claim has been charged by other concurrent transaction.

Case2: Other concurrent transaction which requires the same machine has been validated as non-conflict but has not been committed. The sequence number is only charged when transactions have been committed.

Case3: Other concurrent transaction which requires the same machine has higher importance factor value.

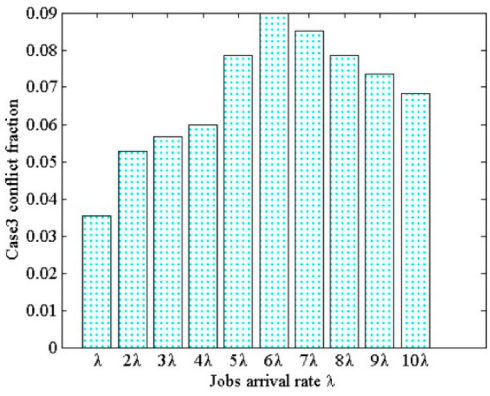


Figure 1. Case3 conflict fraction

In this paper, we mainly explore to optimize tasks in the case3. As Figure 1 shown, in our experiment, less than 0.1 conflicts are in this case. When the jobs arrival rate is 6λ, the proportion of case3 conflicts is relative bigger, so we choose 6λ (1.0) as arrival rate of jobs in our fellow experiments. The other parameters are set up as following:

We use 32 schedulers. The mean duration of tasks is 100.0s, and the mean number of tasks in a job is 100.

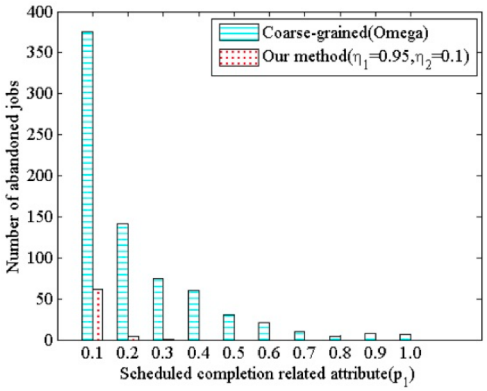


Figure 2. Abandoned jobs number comparison

Figure 2 shows, compare to coarse-grained conflict detection of Omega, our method obviously reduce the abandoned jobs number by giving 0.95 to weight coefficients degrees of scheduled completion rate related attribute p_1.

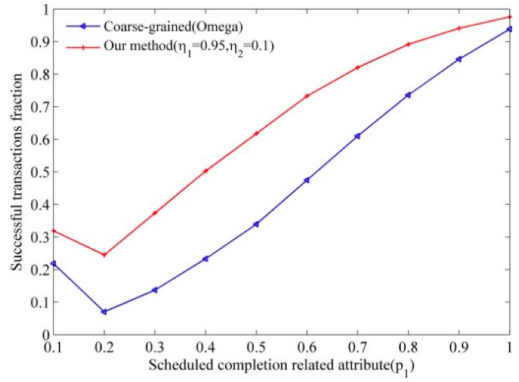


Figure 3. Successful transactions fraction comparison between coarse-grained conflict detection and our method by setting η_1= 0.95 and η_2 =0.1

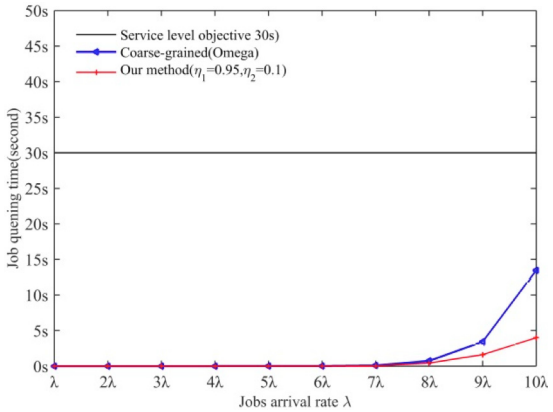
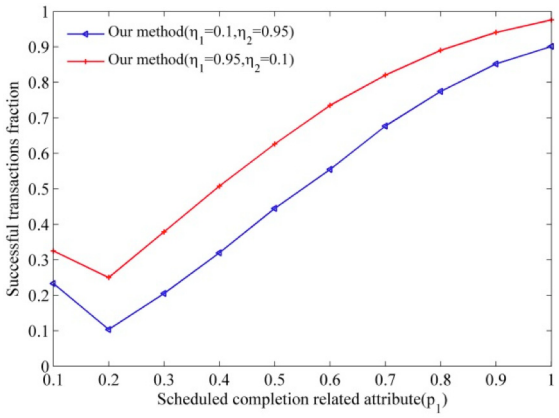


Figure 4. Job queuing time comparison between coarse-grained conflict detection and our method by setting η_1= 0.95 and η_2 =0.1

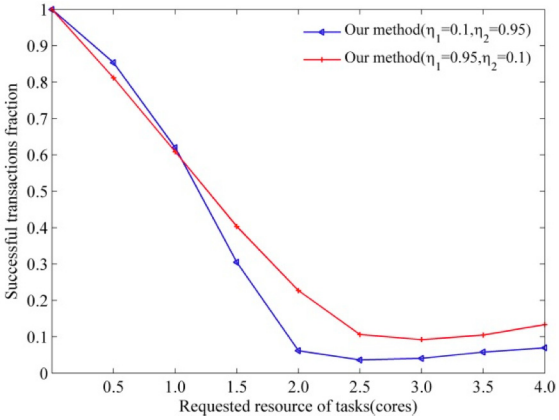
Figure 3 and Figure 4 shows, by using our method, the successful transactions fraction and job queuing time is improved by 2–3 times compare to coarse-grained conflict detection.

This means through D-S fusion method to let tasks of higher scheduled completion become more importance may benefit for optimizing the global scheduling results.

On the other side, combine with the results of Figure 2, Figure 3 and Figure 4, we can tentatively draw the conclusion that our method achieve some optimized scheduling goal rather than coarse-grained conflict detection of Omega.



(a) Successful transactions fraction comparison under different scheduled completion of tasks



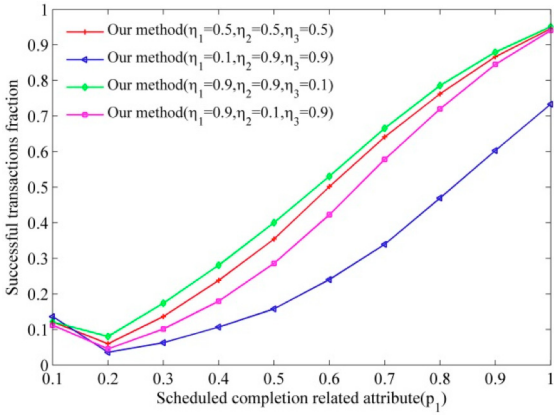
(b)Successful transactions fraction comparison under different requested resource of tasks

Figure 5. Successful transactions fraction comparison by setting two types of η : η_1= 0.95, η_2 =0.1 and η_1= 0.1, η_2 =0.95

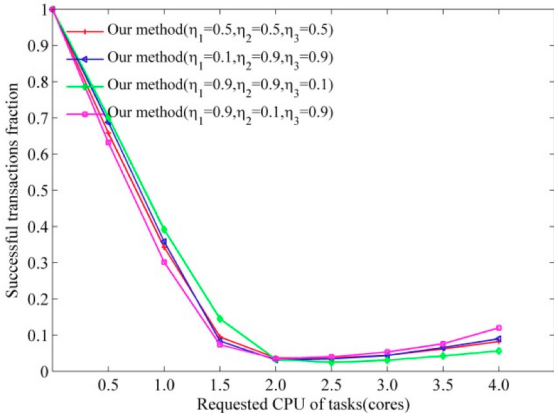
The follow experiments are designed to explore the effect of giving different weight coefficients degrees to different attribute and using different attribute model.

Figure 5(a) demonstrates that the successful transactions fraction by giving the weight coefficients degree $\eta_1 = 0.95$ to scheduled completion related attribute is approximately 1.5 times by giving weight coefficients degree $\eta_1 = 0.1$. Figure 5(b) would seem to show that the successful transactions fraction of tasks who's requested CPU cores bigger than 1.0 by giving the degree $\eta_2 = 0.95$ to p_2 is smaller than by giving the degree $\eta_2 = 0.1$ to p_2. Through our normalized formula, less demand for CPU of a task, the value of p_2 is bigger. Therefore, by giving bigger weight coefficients degree to η_2, the p_2 will more affect the scheduling results.

So, Figure 5 indicates that by giving different weight coefficients and through D-S combination rules, the scheduling results are different. To further verify the scalability of this mechanism, we extend to three attributes in the following experiments.



(a) Successful transactions fraction comparison under different scheduled completion of jobs



(b) Successful transactions fraction comparison under different requested CPU of tasks

Figure 6. Successful transactions fraction comparison by setting four types of η

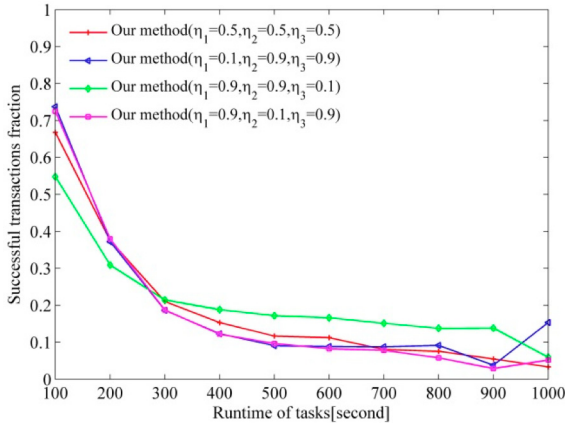


Figure 7. Successful transactions fraction comparison under different runtime of tasks by setting four types of η

Figure 6 and Figure 7 demonstrates that the final successful transactions fraction of scheduling transactions is obviously charged by giving different weight coefficients degree to different attribute. For completion rate related attributep_1, our method performs best compared to other setups we tried when we set $\eta_1 = 0.9, \eta_2 = 0.9, \eta_3 = 0.1$ (Figure 6(a)). For our task requested CPU related scheduling goal, $\eta_1 = 0.9, \eta_2 = 0.9, \eta_3 = 0.1$ is also the best setup among others. As for our duration related scheduling goal, $\eta_1 = 0.9, \eta_2 = 0.1, \eta_3 = 0.9$ is the best. So how to choose attribute and weight coefficients degree is more closely related specific application and worthy further research.

V. CONCLUSIONS AND FUTURE WORK

In this paper, our work explores how to through OCC to achieve some optimized operation for the same production band tasks. So we design and implement multiple attribute D-S evidence theory based OCC in the simulator of Omega and compare to its coarse-grained conflict detection. The evaluation of experiments verifies that it is a feasible and effective method.

Our future work could usefully focus on those things:

In this paper, we explore to use multi attribute and D-S combinational rule in OCC. However, except the scheduled completion related attribute can see obviously practical effect. The effect of other two attributes is hard to evaluate. So, in the future, we will focus on research how to better set attributes and get the proper weight coefficients degrees to let them better service to specific application.

ACKNOWLEDGMENT

We would like to thank the authors of Omega for their selflessness to public the simulator of Omega. This simulator make up for our lack of experimental environment.

REFERENCES

[1] Zaharia, M., Borthakur,D., Sen Sarma,J., Elmeleegy, K., Shenker, S.,and Stoica, I., "Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling", Proc. of the 5th European Conference on Computer Systems(EuroSys'10), ACM, 2010,pp. 265–278. https://doi.org/10.1145/1755913.1755940

[2] Hindman, B., Konwinski, A., Zaharia, M., Ghodsi ,A., Joseph, A. D., Katz, R., Shenker, S. ,and Stoica, I., " Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center", Proc. of the 8th USENIX conference on Networked systems design and implementation(NSDI'11), ACM,2011,pp.295-308.

[3] Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T. , Lowe, J., Shah, H. and Seth S., Saha, B., Curino, C., O'Malley, O., Radia, S. , Reed, B. and Baldeschwieler. E., "Apache Hadoop YARN: yet another resource negotiator", Proc. of the 4th Symposium on Cloud, ACM, 2013, pp.1-16. https://doi.org/10.1145/2523616.2523633

[4] Schwarzkopf, M., Konwinski, A., Abd-El-Malek, M. and Wilkes, J., "Omega: flexible, scalable schedulers for large compute clusters", Proc. of the 8th ACM European Conference on Computer Systems(EuroSys' 13), ACM, 2013, pp.351-364. https://doi.org/10.1145/2465351.2465386

[5] Ousterhout, K., Wendell, P., Zaharia, M., and Stoica., I., "Sparrow: Distributed, low latency scheduling", Proc. of the Twenty-Fourth ACM Symposium on Operating Systems Principles(SOSP□13), ACM, 2013, pp.69–84. https://doi.org/10.1145/2517349.2522716

[6] Eric Boutin, Jaliya Ekanayake, Wei Lin, Bing Shi, Jingren Zhou, Zhengping Qian, Ming Wu, and Lidong Zhou, "Apollo: Scalable and Coordinated Scheduling for Cloud-Scale Computing", Proc. of the 11th USENIX Symposium on Operating Systems Design and Implementation(OSDI '13), ACM, 2013 , pp.285-300.

[7] Verma, A., Pedrosa, L., Abd-El-Malek, M., Korupolu, M., Oppenheimer, D., Tune, E., and Wilkes, J., "Large-scale cluster management at Google with Borg", Proc. the Tenth European Conference on Computer Systems (EuroSys'15), ACM, 2015, pp.18.

[8] Delimitrou, C., Sanchez, D., and Kozyrakis, C., Tarcil: reconciling scheduling speed and quality in large shared clusters, Proc. of the Sixth ACM Symposium on Cloud Computing(SoCC '15), ACM, 2015, pp. 97-110. https://doi.org/10.1145/2806777.2806779

[9] DELGADO, P., DINU, F., KERMARREC, A.-M., AND ZWAENEPOEL, W, "Hawk: Hybrid datacenter scheduling",

Proc. Of 2015 USENIX Annual Technical Conference (USENIX ATC 15),SENIX Association,2015, pp. 499–510.

[10] PAPADIMITRIOU, C. H., "Serializability of concurrent updates", Journal of the ACM, ACM, vol. 26, n.4, 1979, pp.631-653. https://doi.org/10.1145/322154.322158

[11] ESWARAN, K. P., GRAY, J. N., LORIE, R. A., and TRAIGER I. L., "The notions of consistency and predicate locks in a database system", Communications of the ACM,ACM, vol.19, n.11,1976, pp.: 624-633. https://doi.org/10.1145/360363.360369

[12] Kung, H.T., and Robinson, J. T., "On optimistic methods for concurrency control", ACM Transactions on Database Systems (TODS), ACM, vol.6, n.2, 1981, pp.213–226.

[13] BERNSTEIN, P. A., AND GOODMAN, N, "Concurrency control in distributed database systems", ACM Computing Surveys, ACM,vol.13, n.2, 1981, pp.185-221. https://doi.org/10.1145/356842.356846

[14] CHAN, A., FOX, S., LIN, W., NORI, A., AND RIES, D, "The implementation of an integrated concurrency control and recovery scheme", Proc. of the ACM SIGMOD International Conference on Management of Data, ACM, 1982, pp.184-191.

[15] ROBINSON, J, "Design of concurrency controls for transaction processing systems", Ph.D. dissertation, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pa., 1982.

[16] J.R.Haritsa, M.J.Caery, and M.Livny, "Dynamic Real-Time Optimistic Concurrency Control", Proc. of the IEEE Real-Time Systems Symposium, IEEE, 1990, pp. 94-103. https://doi.org/10.1109/real.1990.128734

[17] J.R.Haritsa, M.J.Carey and M.Livny. "On Being Optimistic About Real-Time Constraints", Proc. of the ACM Symposium on Principles of Database Systems, 331-343, Nashville, Tennessee, 1990 https://doi.org/10.1145/298514.298585

[18] REED, D, "Naming and synchronization in a decentralized computer system", Ph.D. dissertation, Dept. of Electrical Engineering and Computer Science, MIT, Cambridge, Mass., 1978.

[19] Yao, J., Wu, C., Xie, X., & Qian, K., "A new method of information decision-making based on D-S evidence theory", Proc. of IEEE International Conference on Systems Man & Cybernetics. IEEE.2010. pp.1804-1811

[20] Li, G., Zou, H., & Yang, F, "Fuzzy ontology and fuzzy d-s evidence theory based context modeling and uncertainty reasoning", Journal of Convergence Information Technology, vol.6, n.12, 2011, pp.185-193. https://doi.org/10.4156/jcit.vol6.issue12.24

[21] Fan, X., & Zuo, M. J,(2006). "Fault diagnosis of machines based on d–s evidence theory part 1: d–s evidence theory and its improvement". Pattern Recognition Letters, vol.27, n.5, 2006, pp.366-376. https://doi.org/10.1016/j.patrec.2005.08.025

[22] .Zadeh, L., "A simple view of the Dempster-Shafer Theory of Evidence and its implication for the rule of combination", The Al Magazine, Vol. 7, N. 2, 1986,pp. 85-90,.

[23] Walley, P., "Statistical reasoning with imprecise probabilities", Applied Statistics,Vol.42, N.42,1993.

[24] CHEN, Y., GANAPATHI, A. S., GRIFFITH, R., and KATZ, R. H., "Design insights for MapReduce from diverse production workloads", Tech. Rep. UCB/EECS–2012–17, UC Berkeley,Jan. 2012.

[25] KAVULYA, S., TAN, J., GANDHI, R., and NARASIMHAN, P., "An analysis of traces from a production MapReduce cluster", Proc. of 10th IEEE/ACM International Conference on (CCGrid'10), IEEE, 2010, pp. 94–103.

[26] REISS, C., TUMANOV, A., GANGER, G. R., KATZ, R. H., and KOZUCH, M. A., Heterogeneity and dynamicity of clouds at scale: Google trace analysis. Proc. of the Third ACM Symposium on Cloud Computing(SoCC'12), ACM, 2012, pp.1-13.

AUTHORS

Libo He with the School of Information Science and Engineering in Yunnan University, No.2 Cuihu North Rd, Kunming City, China (22013000170@mail.ynu.edu.cn)

Zhenping Qiang with the School of Information Science and Engineering in Yunnan University, No.2 Cuihu North Rd, Kunming City, China (qzp@swfu.edu.cn)

Wei Zhou with the School of Software in Yunnan University, No.2 Cuihu North Rd, Kunming City, China (zwei@ynu.edu.cn)

Shaowen Yao with the School of Software in Yunnan University, No.2 Cuihu North Rd, Kunming City, China (yaosw@ynu.edu.cn)

This work is supported by National Natural Science Foundation of China (No. 61363021, No. 61540061). Submitted 26 October 2016. Published as resubmitted by the authorts 27 November 2016.