Supplemental Material: Long-Term Multi-Resource Fairness for Pay-as-you Use Computing Systems

Shanjiang Tang, Zhaojie Niu, Bingsheng He, Bu-Sung Lee, Ce Yu

Abstract—Many current computing systems such as clouds and supercomputers charge users for their resource usages. A user's demand is often changing over time, indicating that it is difficult to keep the high resource utilization all the time for cost efficiency. Resource sharing is a classical and effective approach for high resource utilization. In view of the heterogeneous resource demands of users' workloads, multi-resource allocation fairness is a must for resource sharing in such *pay-as-you-use computing systems*. However, we find that, existing multi-resource fair policies such as Dominant Resource Fairness (DRF), implemented in currently popular resource management systems such as Apache YARN [4] and Mesos [23], are *not* suitable for the pay-as-you-use computing systems. We show that this is because of their *memoryless* characteristic that can cause the following problems in the pay-as-you-use computing systems: 1). users can get resource benefits by cheating; 2). users might not be able to get the total amount of resources that they are entitled to in terms of their resource contributions. In this paper, we propose a new policy called H-MRF, which generalizes DRF and Asset Fairness with the long-term notion. We show that it can address these problems and is suitable for pay-as-you-use computing systems. We have implemented it into YARN by developing a prototype called *MRYARN*. Finally, we evaluate H-MRF using both testbed and simulated experiments. The experimental results show that there are about $1.1 \sim 1.5$ sharing benefit degrees and $1.2\times \sim 1.8\times$ performance improvement for users with H-MRF, better than existing fair schedulers.

Keywords—Long-Term Multi-Resource Fairness, Cloud Computing, SuperComputing, YARN, MRYARN.

APPENDIX A
THE PROPERTIES ANALYSIS FOR LT-DRF

Theorem 1: LT-DRF is truthfulness.

Proof: LT-DRF targets at the fairness of accumulated dominant resources. In LT-DRF, a user can *yield* her unused resources when she does not need. Later when she has more resource demands, LT-DRF will allow her to get an extra amount of resources back from others that she yielded before. It means that the underloaded users are *truly* willing to release unused resources. In contrast, for overloaded users, it is possible for them to cheat the system at a moment in order for obtaining more resources in preemption with others. However, the result of cheating is indeed a pre-consumption of their own resources and they need to *return* at a later time to others. Cheating does not benefit users at all and hence LT-DRF is truthfulness. □

Theorem 2: LT-DRF violates the resource-as-you-contributed fairness property.

Proof: Consider the previous two-user Example 4 of the main file. After t_2, there are totally $50(=\ 15\ +\ 35)$ tasks allocated for A (i.e., A's total resource consumption is <50 CPUs, 100 GB>) and $100(=\ 70\ +\ 30)$ tasks allocated for B (i.e., B's total resource consumption is <100 CPUs, 100 GB>). For each of later successive time step $t_3, t_4, ...$, let us assume that there are more than 25 tasks for A and 50 tasks for B. With LT-DRF, there are exactly 25 tasks allocated for A and 50 tasks allocated for B at each successive time step. Then the total resource consumption will finally be $< (50+25\times\eta)$ CPUs, $(100+50\times\eta)$ GB> for A, and $< (100+50\times\eta)$ CPUs, $(100+50\times\eta)$ GB> for B, where η denotes the number of time steps after t_2. Therefore, the total resource consumption for B is larger than A, violating the resource-as-you-contributed fairness. □

APPENDIX B
THE PROPERTIES ANALYSIS FOR LT-AF

Theorem 3: LT-AF satisfies resource-as-you-contributed fairness property.

Proof: For multi-resource allocation, there are two factors affects resource-as-you-contributed fairness. One is the heterogeneous task requirements between users (e.g., <1 CPU, 2GB> for user A, <1 CPU, 1 GB> for user B in Figure 4 of the main file). Another is the varied workloads (i.e., the number of tasks to run) across users at different time (i.e., *unbalanced* workload which can be either smaller or larger than user's current share). LT-AF overcomes the heterogeneous task demand and varied workload problem by considering the fairness on the basis of aggregate accumulated resource share. It adjusts the current allocation dynamically to each user, to make sure that the aggregate accumulated multiple resources

- *S.J. Tang and C. Yu are with the School of Computer Science&Technology, Tianjin University. E-mail: {tashj, yuce}@tju.edu.cn.*
- *Z.J. Niu and B.S. He are with the School of Computing, National University of Singapore. E-mail: nzjemail@gmail.com. hebs@comp.nus.edu.sg*
- *B.S. Lee is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. E-mail: ebslee@ntu.edu.sg.*

are *close* to each other across users. Therefore, LT-AF meets resource-as-you-contributed fairness property. □

Theorem 4: LT-AF violates sharing incentive property.

Proof: Recall in Section 3.3 of the main file, we show that Asset Fairness is not sharing incentive by giving a counterexample in Section 3.3 of the main file. If we assume that at each time step of resource allocation, the number of tasks pending to run for A is larger than 5 tasks and for B is larger than 15 tasks, respectively. With LT-AF, the allocation in each time will keep the same, i.e., allocating 5 tasks for A and 15 tasks for B, since the aggregate resource shares between A and B are equal at each allocation snapshot. However, for B, there are 20 tasks launched in the exclusively non-sharing partition of <20 CPUs, 20 GB> at each time allocation, better than that in the sharing case. Therefore, LT-AF is not sharing incentive. □

APPENDIX C
THE PROPERTY ANALYSIS FOR H-MRF

Theorem 5: H-MRF satisfies sharing incentive property and resource-as-you-contributed fairness.

Proof: Recall in H-MRF, it detects dynamically the sharing degree β_i for each user. The resources are always first allocated to users under sharing loss (i.e., $\beta_i < 1$) so that all users eventually get sharing benefits in the sharing system over non-sharing one. Thus, H-MRF is sharing incentive. In addition, H-MRF will further try to minimize the difference of aggregate accumulated resources for all users in the case that no user is sharing loss. It implies that H-MRF meets resource-as-you-contributed fairness. □

Theorem 6: (Truthfulness) A user cannot have more amount of resources/tasks allocated in H-MRF by falsely reporting her true demand.

Theorem 7: (Pareto efficiency) A user cannot increase her resources/tasks allocation in H-MRF without decreasing other users' allocation when system resources are fully utilized.

Let's prove Theorem 6 and Theorem 7 together as follows:

Proof: Let $\mathbf{D}_i(t)$ and $N_i(t)$ be the resource demand vector and the number of tasks allocated for user i at time t, respectively. Let $a_{i,k}$ denote the task demand of resource k for user i. Then we have $u_{i,k}(t) = N_i(t) \times a_{i,k}$, where $u_{i,k}(t)$ denotes the amount of resource k allocated for user i with H-MRF at time t. Moreover, the resource utilization of the system is fully maximized (i.e., no more tasks can be allocated), assuming there are an unbounded number of tasks at time t. In that case, user i must have one bottleneck resource k, otherwise the system will not be fully utilized. Then we have the resulting allocation for all users w.r.t resource k satisfying $\sum_{1\leqslant j\leqslant n \wedge j\neq i} u_{j,k}(t) + u_{i,k}(t) \leqslant r_k < \sum_{1\leqslant j\leqslant n \wedge j\neq i} u_{j,k}(t) + u_{i,k}(t) + a_{i,k}$, where r_k denotes the system capacity of resource k. We now come to prove Theorem 6 and Theorem 7 in the following:

Proof of Theorem 6: Let's start the proof by contradiction to assume user i can have more resources/tasks allocated by using a resource demand vector $\mathbf{D}_i^{'}(t)$, i.e., $\mathbf{D}_i^{'}(t) \neq \mathbf{D}_i(t) \Rightarrow N_i^{'}(t) > N_i(t) \Leftrightarrow N_i^{'}(t) \geqslant N_i(t) + 1$, where $N_i^{'}(t)$ denotes the number of tasks allocated for user i by using $\mathbf{D}_i^{'}(t)$. Then it holds that $u_{i,k}^{'}(t) = N_i^{'}(t) \times a_{i,k} \geqslant (N_i(t) + 1) \times a_{i,k} = u_{i,k}(t) + a_{i,k}$ for every resource k, where $u_{i,k}^{'}(t)$ represents the corresponding amount of resource k allocated for user i using $\mathbf{D}_i^{'}(t)$. Then the allocation for resource k in the system will be $\sum_{1\leqslant j\leqslant n \wedge j\neq i} u_{j,k}(t) + u_{i,k}^{'}(t) \geqslant \sum_{1\leqslant j\leqslant n \wedge j\neq i} u_{j,k}(t) + u_{i,k}(t) + a_{i,k} > r_k$, being out of the system capacity and thus it is a false allocation. In other words, the assumption is wrong and thus H-MRF is truthfulness.

Proof of Theorem 7: Assume we can increase the number of tasks from $N_i(t)$ to $N_i^{'}(t)$ (i.e., $N_i^{'}(t) \geqslant N_i(t) + 1$) without decreasing other users' allocation. Then it holds $u_{i,k}^{'}(t) \geqslant u_{i,k}(t) + a_{i,k}$, where $u_{i,k}^{'}(t)$ represents the corresponding amount of resource k allocated for $N_i^{'}(t)$ tasks of user i. Given $N_i^{'}(t)$ for user i, we then have a new resulting allocation for resource k satisfying $\sum_{1\leqslant j\leqslant n \wedge j\neq i} u_{j,k}(t) + u_{i,k}^{'}(t) \geqslant \sum_{1\leqslant j\leqslant n \wedge j\neq i} u_{j,k}(t) + u_{j,k}(t) + a_{i,k} > r_k$, which contradicts our hypothesis and thus our proof completes. □

APPENDIX D
H-MRF EXTENSION FOR DISTRIBUTED RESOURCE ALLOCATION

In previous sections, we have implicitly assumed one *'super-machine'* system containing all big resources. However, in practice, it is more likely to have a cluster consisting of many small distributed computing nodes. In the following, we turn to see how H-MRF can be adapted to use in such distributed scenario.

Consider a cluster consisting of K computing nodes. Let $\mathbf{R}^{(h)} =< r_1^{(h)}, ..., r_m^{(h)} >$ be the total resource capacity for the h^{th} machine (i.e., $\mathbf{R} = \sum_{1\leqslant h\leqslant K} \mathbf{R}^{(h)}$), and $\mathbf{S}_i^{(h)} =< s_{i,1}^{(h)}, ..., s_{i,m}^{(h)} >$ be the current resource share for user i in the h^{th} machine (i.e., $\mathbf{S}_i = \sum_{1\leqslant h\leqslant K} \mathbf{S}_i^{(h)}$). Moreover, let $u_{i,j}^{(h)}(t)$ denote the current allocated resource for user i in the h^{th} machine with respect to resource type j (i.e., $u_{i,j}(t) = \sum_{1\leqslant h\leqslant K} u_{i,j}^{(h)}(t)$). In such a distributed environment, instead of separately performing resource allocation for each machine, we use H-MRF to jointly consider the resource allocation across all machines. The detailed process is as follows. Moreover, let $\beta_i^{(h)}$ denote the local sharing degree for user i in the h^{th} machine (i.e., $\beta_i = \sum_{1\leqslant h\leqslant K} \beta_i^{(h)}$). For each allocation in the h^{th} machine, we first compute the *global* sharing degree β_i for each user i according to Formula 2 and 3 of the main file as follows: $\beta_i = \min_{1\leqslant j\leqslant m} \beta_{i,j}(t) = \frac{\int_0^t u_{i,j}(t)\,\mathbf{d}t}{\int_0^t \min\{d_{i,j}(t), s_{i,j}(t)\}\,\mathbf{d}t} = \frac{\int_0^t \sum_{1\leqslant h\leqslant K} u_{i,j}^{(h)}(t)\,\mathbf{d}t}{\int_0^t \min\{d_{i,j}(t), s_{i,j}^{(h)}(t)\}\,\mathbf{d}t}$. Then we can take Algorithm 1 of H-MRF in the main file to perform the resource allocation for the h^{th} machine. We particularly show that it meets all desired properties listed in Section 2 of the main file for distributed resource allocation. Moreover, let $N_i^{(h)}(t)$ denote the number of tasks allocated for user i in the h^{th} machine (i.e., $N_i(t) = \sum_{1\leqslant h\leqslant K} N_i^{(h)}(t)$).

Theorem 8: In the distributed scenario, H-MRF satisfies sharing incentive property and resource-as-you-contributed fairness.

Proof: In the distributed scenario, H-MRF jointly performs resource allocation across machines based on the global

sharing degree β_i for each user i. The sharing-loss users (i.e., $\beta_i < 1$) are always given higher priorities than the sharing-benefit users (i.e.,$\beta_i \geqslant 1$) such that all users get sharing benefits over the non-sharing case in the end. Therefore, H-MRF satisfies sharing incentive property. Furthermore, H-MRF minimizes the aggregate accumulated resource difference across all users given that all users are sharing-benefit. It indicates that H-MRF satisfies resource-as-you-contributed fairness. □

Theorem 9: H-MRF satisfies both truthfulness and pareto efficiency properties in the distributed resource allocation.

Proof: Let $N_i^{(h)}(t)$ be the number of tasks allocated for user i in the h^{th} machine at time t (i.e., $N_i^{'}(t) = \sum_{1 \leqslant h \leqslant K} N_i^{(h)}(t)$). It then holds $u_{i,k}^{(h)}(t) = N_i^{(h)}(t) \times a_{i,k}$. Moreover, each machine of the system is fully utilized (i.e., no more tasks can be allocated), with the assumption that there are an unbounded number of tasks at time t. Given that, there must be one bottleneck resource k for each user i for each machine, otherwise it cannot be fully utilized for that machine. It then holds for resource type k that $\sum_{1 \leqslant j \leqslant n \wedge j \neq i} u_{j,k}^{(h)}(t) + u_{i,k}^{(h)}(t) \leqslant r_k^{(h)} < \sum_{1 \leqslant j \leqslant n \wedge j \neq i} u_{j,k}^{(h)}(t) + u_{i,k}^{(h)}(t) + a_{i,k}$.

1). Truthfulness Proof: By contradiction, let's assume that user i can allocate more resources/tasks by using a new resource demand vector $\mathbf{D}_i^{'}(t)$, i.e., $\mathbf{D}_i^{'}(t) \neq \mathbf{D}_i(t) \Rightarrow N_i^{'}(t) > N_i(t)$, where $N_i^{'}(t)$ denotes the number of tasks allocated for user i by using $\mathbf{D}_i^{'}(t)$. There must be true for at least a machine h that $N_i^{'(h)}(t) > N_i^{(h)}(t) \Leftrightarrow N_i^{'(h)}(t) \geqslant N_i^{(h)}(t) + 1$, where $N_i^{'(h)}(t)$ denotes the number of tasks allocated for user i by using $\mathbf{D}_i^{'}(t)$ in the h^{th} machine. We then have $u_{i,k}^{'(h)}(t) = N_i^{'(h)}(t) \times a_{i,k} \geqslant (N_i^{(h)}(t) + 1) \times a_{i,k} = u_{i,k}^{(h)}(t) + a_{i,k}$ for every resource k, where $u_{i,k}^{'(h)}(t)$ represents the corresponding amount of resource k allocated for user i using $\mathbf{D}_i^{'}(t)$ in the h^{th} machine. It thereby holds for resource k in the h^{th} machine that $\sum_{1 \leqslant j \leqslant n \wedge j \neq i} u_{j,k}^{(h)}(t) + u_{i,k}^{'(h)}(t) \geqslant \sum_{1 \leqslant j \leqslant n \wedge j \neq i} u_{j,k}^{(h)}(t) + u_{i,k}^{(h)}(t) + a_{i,k} > r_k^{(h)}$, exceeding the capacity of the h^{th} machine and hence the proposition is false. That is, H-MRF is truthfulness in the distributed resource allocation.

2). Pareto Efficiency Proof: Let's by contradiction suppose that The number of tasks can be increased from $N_i(t)$ to $N_i^{'}(t)$ (i.e., $N_i^{'}(t) \geqslant N_i(t) + 1$) without decreasing other users' allocation. It must hold for at least one machine h that satisfies $N_i^{'(h)}(t) \geqslant N_i^{(h)}(t) + 1$. Therefore we have $u_{i,k}^{'(h)}(t) \geqslant u_{i,k}^{(h)}(t) + a_{i,k}$, where $u_{i,k}^{'(h)}(t)$ denotes the corresponding amount of resource k for $N_i^{'(h)}(t)$ tasks of user i. Therefore we have $\sum_{1 \leqslant j \leqslant n \wedge j \neq i} u_{j,k}^{(h)}(t) + u_{i,k}^{'(h)}(t) \geqslant \sum_{1 \leqslant j \leqslant n \wedge j \neq i} u_{j,k}^{(h)}(t) + u_{j,k}^{(h)}(t) + a_{i,k} > r_k^{(h)}$, being out of the capacity of the h^{th} machine. It means that the hypothesis is wrong and therefore H-MRF is pareto efficiency. □

APPENDIX E
MONETARY COST EVALUATION

Recall in Section 3 of the main file that Amazon EC2 offers three kinds of pricing schemes (e.g., on-demand, reserved and spot instances) for users. Our MRYARN system supports users to contribute their instances of either *different* or the *same* pricing schemes to the resource pool and share together with others. It then guarantees the resource-as-you-contribute resource fairness across users. Particularly, when a user's tasks complete, she can leave the system at arbitrary time by stopping her own instances and Amazon EC2 charges her instances according to her selected pricing scheme and running time.

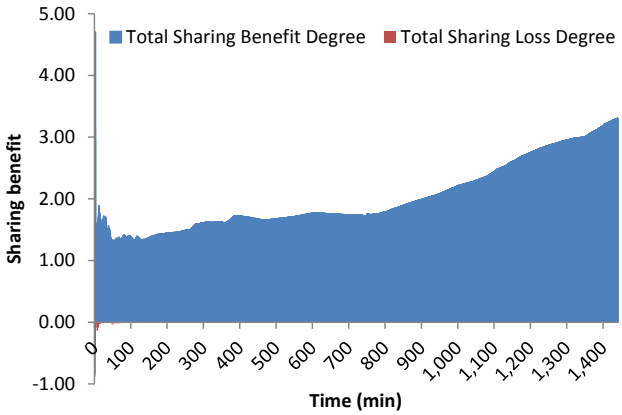
Table 1 shows monetary cost results for four users under different resource allocation policies and pricing schemes, where the on-demand, reserved and observed spot prices for m3.xlarge instances are \$0.266/h, \$0.190/h and \$0.0508/h, respectively. Moreover, the Amazon EC2 charges users on a *per-second* basis for m3.xlarge instances. First, users under static partitioning pay the most for instances compared with other policies in the same pricing scheme. For example, there can be as large as 125% monetary cost saving for H-MRF compared to the static partitioning (non-sharing) case for User 1. This is because resource sharing can be better than static partitioning in performance for users as illustrated in Figure 9 of the main file. Second, H-MRF outperforms other policies in monetary cost savings under all pricing schemes for users, since it achieves better performance than others as discussed in Section 7.2.3 of the main file. Third, the difference of monetary costs under different pricing schemes for a user is significant. For example, there can be 40% cost saving for reserved instances over on-demand instances for User 1 under H-MRF. It indicates that choosing appropriate pricing scheme and scheduling policy are crucial for monetary cost saving maximization in the pay-as-you-use computing system.

		On-demand	Reserved Instance	Spot Instance
User 1 (Facebook)	Static Partitioning	4.1895	2.9925	0.8001
	DRF	2.128	1.52	0.4064
	LT-DRF	2.5935	1.8525	0.4953
	LT-AF	2.5935	1.8525	0.4953
	H-MRF	**1.862**	**1.33**	**0.3556**
User 2 (Purdue)	Static Partitioning	2.5935	1.8525	0.4953
	DRF	1.3965	0.9975	0.2667
	LT-DRF	1.729	1.235	0.3302
	LT-AF	1.862	1.33	0.3556
	H-MRF	**1.197**	**0.855**	**0.2286**
User 3 (Spark)	Static Partitioning	4.9875	3.5625	0.9525
	DRF	4.655	3.325	0.889
	LT-DRF	4.389	3.135	0.8382
	LT-AF	4.3225	3.0875	0.8255
	H-MRF	**3.591**	**2.565**	**0.6858**
User 4 (TPC-H)	Static Partitioning	5.852	4.18	1.1176
	DRF	4.655	3.325	0.889
	LT-DRF	4.921	3.515	0.9398
	LT-AF	4.5885	3.27755	0.8763
	H-MRF	**3.5245**	**2.5175**	**0.6731**

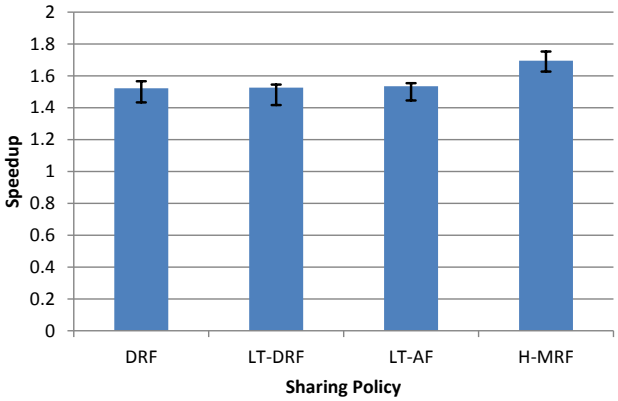
TABLE 1: The monetary cost for users under different allocation policies and pricing schemes, where the on-demand, (1-year term) reserved and spot prices for m3.xlarge instance are \$0.266/h, \$0.190/h and \$0.0508/h, respectively. Users pay for instances by per second.

APPENDIX F
GOOGLE TRACE DRIVEN SIMULATION RESULTS

In this section, we start to evaluate H-MRF at a larger scale by using our built simulator to replay the execution of tasks from



(a) Total sharing benefit/loss degree for users under H-MRF policy.



(b) Performance results for different fair sharing policies.

Fig. 1: Trace-driven simulation results for users submitting tasks in 24 hours to a shared Google cluster of 1000 machines.

a Google cluster with Google cluster-usage traces [8]. Figure 1 of the main file shows the distribution for tasks w.r.t different resource demands. In the following, we simulate 100 users submitting tasks with different resource demands for *three* resource types (i.e., CPU, memory, and disk) in 24 hours to a Google cluster of 1000 machines, consisting of 527.5 CPU units, 513.2 memory units and 520.0 disk units in total, based on the Google trace's provided capacity information about machines. We assume that users share the cluster with equal share to each other.

Recall in Section 4 of the main file that for user i, it is always *true* for sharing degree $\beta_i(t) = 1$ in the *non-shared* partitioned environment. Thus, to show how much better/worse for all users in the *shared* system compared to the non-shared partitioned environment, we introduce two concepts, namely, *total sharing benefit degree* and *total sharing loss degree* for all users, where *total sharing benefit degree* is defined as $\sum_{i=1}^{n} \max\{\beta_i(t) - 1, 0\}$, and the *total sharing loss degree* is computed by $\sum_{i=1}^{n} \min\{\beta_i(t) - 1, 0\}$. Figure 1(a) presents the total sharing benefit/loss degrees for users under H-MRF policy. In most scenarios, users are getting sharing benefits under the shared Google cluster with H-MRF over a non-shared partitioned cluster except the beginning computation. Regarding some of the sharing loss at the beginning, it is due to the unavoidable resource waiting problem for later arriving users, referred to Section 7.2.2 of the main file for the detailed analysis. Although this problem causes sharing loss for some users, our H-MRF policy can address it via enabling those users to have a higher priority in next time resource allocation and makes all users get sharing benefits finally.

Figure 1(b) illustrates the speedup performance results for different fair policies, where *speedup* is defined as the ratio of execution time under the non-shared static partitioned cluster to that of the shared cluster. For each error bar, it illustrates the average, minimum and maximum performance results for all users' workloads under a fair policy. There are about on average $1.5\times \sim 1.7\times$ performance improvement for users' workloads in the shared environment (e.g., DRF, LT-DRF, LT-AF, H-MRF) compared to the exclusively non-shared computation. Furthermore, H-MRF outperforms other alternative policies in performance due to its efficient task placement.

In summary, all of these observation results are consistent with those in Section 7.2.2 and Section 7.2.3 of the main file.

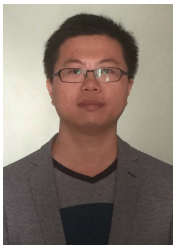
ACKNOWLEDGEMENT

This work is sponsored by the National Natural Science Foundation of China (61602336,61772544,U1731125). Ce Yu's work is supported by the National Natural Science Foundation of China (U1531111,U1731243). Bingsheng and Zhaojie's work is supported by a MoE AcRF Tier 1 grant (T1 251RES1610) and an NUS startup grant in Singapore.

REFERENCES

[1] Apache hadoop. In *http://hadoop.apache.org*.
[2] Apache hive performance benchmarks. In *https://issues.apache.org/jira/browse/HIVE-396*.
[3] Apache tpc-h benchmark on hive. In *https://issues.apache.org/jira/browse/HIVE-600*.
[4] Apache yarn. In *https://hadoop.apache.org/docs/current2/index.html*.
[5] Blue waters. In *http://www.ncsa.illinois.edu/enabling/bluewaters/*.
[6] Comet. In *http://www.sdsc.edu/services/hpc/hpc_systems.html*.
[7] Facebook workload traces. In *https://github.com/SWIMProjectUCB/SWIM/wiki/Workloads-repository*.
[8] Google cluster data. In *https://code.google.com/p/googleclusterdata/*.
[9] Max-min fairness (wikipedia). In *http://en.wikipedia.org/wiki/Max-min_fairness*.
[10] Mira. In *http://www.alcf.anl.gov/mira*.
[11] Puma datasets. In *http://web.ics.purdue.edu/~fahmad/datasets.htm*.
[12] May Al-Roomi, Shaikha Al-Ebrahim, Sabika Buqrais and Imtiaz Ahmad. Cloud Computing Pricing Models: A Survey. *International Journal of Grid and Distributed Computing*, 6(5):93–106, 2013.
[13] Faraz Ahmad, Seyong Lee, Mithuna Thottethodi, and T. N. Vijaykumar. Puma: Purdue mapreduce benchmarks suite. In *ECE Technical Reports*, 2012.
[14] Arka A. Bhattacharya, David Culler, Eric Friedman, Ali Ghodsi, Scott Shenker, and Ion Stoica. Hierarchical scheduling for diverse datacenter workloads. In *SOCC '13*, pages 4:1–4:15, NY, USA, 2013. ACM.
[15] Alex D. Breslow, Ananta Tiwari, Martin Schulz, Laura Carrington, Lingjia Tang, and Jason Mars. Enabling fair pricing on hpc systems with node sharing. In *SC '13*, pages 37:1–37:12, NY, USA, 2013. ACM.
[16] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI'04*, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[17] Christina Delimitrou and Christos Kozyrakis. Quasar: Resource-efficient and qos-aware cluster management. In *ASPLOS '14*, pages 127–144, NY, USA, 2014. ACM.
[18] Danny Dolev, Dror G. Feitelson, Joseph Y. Halpern, Raz Kupferman, and Nathan Linial. No justified complaints: On fair sharing of multiple resources. In *ITCS '12*, pages 68–75, NY, USA, 2012. ACM.
[19] Ali Ghodsi, Vyas Sekar, Matei Zaharia, and Ion Stoica. Multi-resource fair queueing for packet processing. In *SIGCOMM '12*, pages 1–12, NY, USA, 2012. ACM.
[20] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *NSDI'11*, pages 323–336, Berkeley, CA, USA, 2011. USENIX Association.
[21] Ali Ghodsi, Matei Zaharia, Scott Shenker, and Ion Stoica. Choosy: Max-min fair sharing for datacenter jobs with constraints. In *EuroSys '13*, pages 365–378, NY, USA, 2013. ACM.
[22] Robert Grandl, Ganesh Ananthanarayanan, Srikanth Kandula, Sriram Rao, and Aditya Akella. Multi-resource packing for cluster schedulers. In *SIGCOMM '14*, pages 455–466, NY, USA, 2014. ACM.
[23] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, and Ion Stoica. Mesos: A platform for fine-grained resource sharing in the data center. In *NSDI'11*, pages 295–308, Berkeley, CA, USA, 2011. USENIX Association.
[24] Michael Isard, Vijayan Prabhakaran, Jon Currey, Udi Wieder, Kunal Talwar, and Andrew Goldberg. Quincy: Fair scheduling for distributed computing clusters. In *SOSP '09*, pages 261–276, NY, USA, 2009. ACM.
[25] Carlee Joe-Wong, Soumya Sen, Tian Lan, and Mung Chiang. Multi-resource allocation: Fairness-efficiency tradeoffs in a unifying framework. *IEEE/ACM Trans. Netw.*, 21(6):1785–1798, December 2013.
[26] Ian Kash, Ariel D. Procaccia, and Nisarg Shah. No agent left behind: Dynamic fair division of multiple resources. In *AAMAS '13*, pages 351–358, 2013.
[27] Haikun Liu and Bingsheng He. Reciprocal resource fairness: Towards cooperative multiple-resource fair sharing in iaas clouds. In *SC '14*, pages 970–981, Piscataway, NJ, USA, 2014. IEEE Press.
[28] David C. Parkes, Ariel D. Procaccia, and Nisarg Shah. Beyond dominant resource fairness: Extensions, limitations, and indivisibilities. *ACM Trans. Econ. Comput.*, 3(1):3:1–3:22, March 2015.
[29] Jubaraj Sahu and Karen R. Heavey. Advanced computational fluid dynamics simulations of projectiles with flow control. In *SC '04*, pages 27–, Washington, DC, USA, 2004. IEEE Computer Society.
[30] Shanjiang Tang, BingSheng He, Shuhao Zhang, and Zhaojie Niu. Elastic multi-resource fairness: Balancing fairness and efficiency in coupled cpu-gpu architectures. In *SC '16*, pages 75:1–75:12, Piscataway, NJ, USA, 2016. IEEE Press.
[31] Shanjiang Tang, Bu-sung Lee, Bingsheng He, and Haikun Liu. Long-term resource fairness: Towards economic fairness on pay-as-you-use computing systems. In *ICS '14*, pages 251–260, NY, USA, 2014. ACM.
[32] Shanjiang Tang, Bu-Sung Lee, and Bingsheng He. Fair Resource Allocation for Data-Intensive Computing in the Cloud. *TSC'18*, 99(PrePrints):1, 2018.
[33] Shanjiang Tang, Bu-Sung Lee, and Bingsheng He. Dynamic Job Ordering and Slot Configurations for MapReduce Workloads. *TSC'16*, vol. 9, no.1, pp.4-17, Jan. 2016.
[34] Shanjiang Tang, Bu-Sung Lee, and Bingsheng He. DynamicMR: A Dynamic Slot Allocation Optimization Framework for MapReduce Clusters. *TCC'16*, vol. 2, no.3, pp.333-347, Oct. 2014.
[35] A. Thusoo, J.S. Sarma, N. Jain, Zheng Shao, P. Chakka, N. Zhang, S. Antony, Hao Liu, and R. Murthy. Hive - a petabyte scale data warehouse using hadoop. In *ICDE'10*, pages 996–1005, March 2010.
[36] Wei Wang, Baochun Li, and Ben Liang. Dominant resource fairness in cloud computing systems with heterogeneous servers. In *INFOCOM'14*, pages 583–591, April 2014.
[37] Wei Wang, Baochun Li, Ben Liang, and Jun Li. Multi-resource fair sharing for datacenter jobs with placement constraints. In *SC '16*, pages 86:1–86:12, Piscataway, NJ, USA, 2016. IEEE Press.
[38] Phil Webster. Nasa center for climate simulation: Data-centric climate computing. SC'11, 2011.
[39] Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, and Ion Stoica. Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling. In *EuroSys '10*, pages 265–278, NY, USA, 2010. ACM.
[40] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *HotCloud'10*, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association.
[41] Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz, and Ion Stoica. Improving mapreduce performance in heterogeneous environments. In *OSDI'08*, pages 29–42, 2008.
[42] Seyed Majid Zahedi and Benjamin C. Lee. Ref: Resource elasticity fairness with sharing incentives for multiprocessors. In *ASPLOS '14*, pages 145–160, NY, USA, 2014. ACM.



Shanjiang Tang received the PhD degree from School of Computer Science and Engineering, Nanyang Technological University, Singapore in 2015, and the MS and BS degrees from Tianjin University, China, in Jan 2011 and July 2008, respectively. He is currently an assistant professor in School of Computer Science and Technology, Tianjin University, China. His research interests include parallel computing, cloud computing, big data analysis, and computational biology.

Zhaojie Niu received his bachelor's degree and master's degree from Huazhong University of Science and Technology(HUST), China, in Jan 2009 and July 2012 respectively. He received his PhD degree from Nanyang Technological University (NTU), Singapore in Sep 2017. He is a research associate in National University of Singapore (NUS). He is interested in resource management, job scheduling and data processing in large-scale clusters.

Bingsheng He received the bachelor's degree in computer science from Shanghai Jiao Tong University in 1999 to 2003, and the PhD degree in computer science from the Hong Kong University of Science and Technology in 2003 to 2008. He is an associate professor in the School of Computing, National University of Singapore. His research interests include high performance computing, distributed and parallel systems, and database systems.

Bu-Sung Lee received the B.Sc. (Hons.) and Ph.D. degrees from the Electrical and Electronics Department, Loughborough University of Technology, U.K., in 1982 and 1987, respectively. He is currently Associate Professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests include computer networks protocols, distributed computing, network management and Grid/Cloud computing.

Ce Yu received his P.h.D degree of Computer Science from Tianjin University(TJU) in 2009, MS, BS degree in 2005 and 1998 in the same University respectly. He is currently an associate professor and director of High Performance Computing Lab(HPCL) of Computer Science& Technology in Tianjin University. His main research interests include High Performance Computing, Big Data, Astro-informatics, Cloud Computing.