IEEE COMMUNICATIONS LETTERS, VOL. 23, NO. 1, JANUARY 2019

Low Complexity Hierarchical Scheduling for Diverse Datacenter Jobs

Chaoqun You, *Student Member, IEEE*, Yang Wang, *Student Member, IEEE*, Shizhong Xu, *Member, IEEE*, Long Luo, *Student Member, IEEE*, Meng-Hsi Chen, *Student Member, IEEE*, and Le Min Li

Abstract—**Datacenter jobs in a private cloud are often generated by different departments or groups of a business or other organization. In order to reflect the organizational structures of datacenter workloads, it is desirable for the datacenter scheduler to support hierarchical scheduling. The state-of-the-art hierarchical scheduling scheme, hierarchical dominant resource fairness (H-DRF), however, is expensive to implement. Specifically, the worst-case time complexity of H-DRF is $O(q^2)$, where q is dominated by the number of jobs in the hierarchy, which is a large complexity in a practical large-scale datacenter. In this letter, we design a new online hierarchical fair scheduling scheme, multi-resource collapsed hierarchies (MCH), in the way of converting the hierarchy into a weighted flat tree. MCH takes at most $O(q)$ time and is simple enough to implement in practice. Simulations driven by Google cluster traces show that MCH consumes a nearly sqrt of the total run time of H-DRF, improving significantly in saving the computation time.**

Index Terms—**Low complexity, hierarchical scheduling, fairness.**

I. Introduction

PRIVATE clouds refer to internal datacenters of a business or other organization. Datacenter jobs in private clouds come from different departments in an organization and are therefore grouped according to the structure of the organization. Meanwhile, datacenter jobs are characterized by a high degree of demand heterogeneity across *multiple resource types* [1], [2]. Workload analyses in datacenters have confirmed that jobs may require vastly different amounts of resources of memory, CPU cores, storage space, I/O bandwidth, *etc* [1]–[6]. Therefore, efficient scheduling in datacenters requires taking multiple resource types into account. As the workloads surge, providing Quality-of-Service guarantees to jobs that are both grouped and heterogenous is increasingly important. The following features are desired for the datacenter scheduler to achieve this objective.

A. Hierarchical Scheduling

Hierarchical scheduling enables scheduling resources to reflect organizational structures of datacenter workloads [3]. As a simple example (see Fig. 1), the ads and dev departments are supposed to be assigned 60% and 40% of the total resources, respectively. Resources are further allocated, 90% to the prod jobs and 10% to the test jobs, in each of the departments.

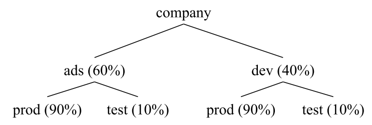


Fig. 1. Example of hierarchical scheduling

B. Fairness

Fair sharing is a fundamental tool to provide *performance guarantee* for datacenter jobs. In a datacenter, a job or a department is supposed to be guaranteed to receive a share irrespective of the demands of its sibling nodes. In a hierarchy, the share guaranteed to a node is referred to as *hierarchical share guarantee* [3].

C. Low Complexity

In datacenters, schedulers need to make *online* decisions frequently. Typical datacenter schedulers, including Hadoop [7] and Mesos [8], make thousands of scheduling decisions per second. Therefore, it is critical for the schedulers to make decisions at high speeds. This requires the scheduling algorithm to have a low time complexity.

The current Hadoop implementations [7], though support hierarchies by turning to a natural adaptation of the original *dominant resource fairness* (DRF) [1], a compelling multi-resource fair scheduler, to the hierarchical setting, they suffer from starvations of some jobs [3]. The recent work, H-DRF [3], avoids the starvation, thus providing hierarchial share guarantees. However, H-DRF has a time complexity of $O(q^2)$, where q is the number of nodes in a hierarchy. As we shall show in Section IV-B, H-DRF incurs a large computation delay for a datacenter with a large q, which is unbearable for delay-sensitive applications.

In this letter, we propose a new online multi-resource scheduling scheme in Section III, Multi-resource Collapsed Hierarchies (MCH), that achieves all the three desirable features. MCH firstly converts a hierarchy into a weighted flat tree and then allocates resources as a weighted DRF scheduler. Although the idea of flattening a hierarchical tree is not new, prior works (*e.g.*, Collapsed Hierarchies (CH) [9]) only work well for a single resource type, naive extensions of CH to schedule multiple resources may *violate* the hierarchical share guarantee [3] because of the *dove-tailing* of jobs' demands, *i.e.*, that jobs have complementary demands (*e.g.*, job 1's $\langle 1 \text{ CPU}, 0 \text{ Mem}\rangle$ and job 2's $\langle 0 \text{ CPUs}, 1 \text{ Mem}\rangle$) "punishes" the parent nodes. In order to provide hierarchical share guarantee to jobs with multiple resource demands, we firstly classify jobs (*i.e.*, leaf nodes) into two categories: *full-type* jobs that require all types of resources; and *partial-type* jobs that require

Manuscript received October 11, 2018; accepted November 7, 2018. Date of publication November 20, 2018; date of current version January 8, 2019. This work is partially supported by NSFC Fund (61671130,61271165,61301153), The National Basic Research Program (China's 973 program) (2013CB329103). The associate editor coordinating the review of this paper and approving it for publication was T. Han. *(Corresponding author: Shizhong Xu.)*

C. You, Y. Wang, S. Xu, L. Luo, and L. M. Li are with the Key Laboratory of Optical Fiber Sensing and Communications, Ministry of Education, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: chaoqunyou@gmail.com; wyatuestc@gmail.com; xsz@uestc.edu.cn; longvslong@gmail.com; lml@uestc.edu.cn).

M.-H. Chen is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 1A1, Canada (e-mail: mchen@ece.utoronto.ca).

Digital Object Identifier 10.1109/LCOMM.2018.2882380

only one or several types of resources. Then, for each category of jobs, we present a formula to compute the corresponding weights for each job if the hierarchy was flattened. As it is shown in the simulations, MCH improves significantly in saving the computation time by generally consuming almost sqrt of the total run time of H-DRF.

II. System Model and Problem Formulation

A. Notations

A hierarchical scheduler is configured with a weighted tree (see Fig. 1). The leaves in the tree denote the jobs that ultimately to be allocated resources, whereas the internal nodes represent the departments or groups. Formally, we use n_i to denote a node in a hierarchy, where i is a list of numbers that describe how the node can be found when starting at the top of the tree and going down, left to right. For example, $n_{2,1,4}$ is the fourth child of $n_{2,1}$. The root node of the hierarchy is denoted as n_r. Let $P(i)$ be the parent node of n_i, and $C(i)$ be the set of active children of n_i. Each node n_i is associated with a weight ω_i. Therefore, the hierarchical share guarantee of node i is $\frac{\omega_i}{\sum_{j \in C(P(i))} \omega_j}$. In this letter, for simplicity, we assume that all nodes in the hierarchy have the same weight.

Let $\mathcal{R} = \{1, 2, \ldots, m\}$ be the set of m resources provided by the datacenter. Let $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ be the datacenter resource capacity vector, where c_j denotes the total amount of resource j. Let $\mathcal{N} = \{1, 2, \ldots, p\}$ be set of p jobs sharing the datacenter. Each job submits a set of identical tasks. Let d_i^j be the amount of resource j ($j \in \mathcal{R}$) required by each task of job n_i and t_i be the number of tasks allocated to n_i. Then for each type of resources $j \in \mathcal{R}$ the *capacity constraint* is denoted as $\sum_{i \in \mathcal{N}} d_i^j t_i \le c_j$.

B. Design Objectives and Constraints

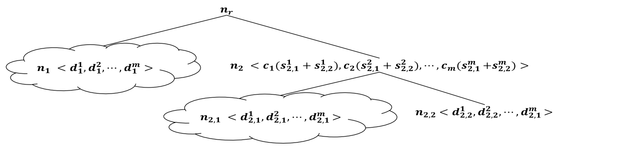
1) Hierarchical Dominant Resource Fairness: Traditionally, for a non-hierarchical scheduling system, a notable algorithm, Dominant Resource Fairness (DRF), was proposed as a promising notion of fairness for multi-resource settings [1]. DRF seeks a *max-min fairness* on each job's dominant share, defined as the fraction of the dominant resource the job has been allocated. Let $\mu_i = \max_j \{\frac{d_{ij}}{c_j}\}$ be the dominant share of a resource required by job i to process one task. Then $\mu_i t_i$ is the dominant share of n_i and the objective of DRF is $\mu_i t_i = \mu_j t_j$, where $i \ne j$, and $i, j \in \mathcal{N}$. Now we are able to extend this objective to hierarchical scheduling as follows,

$$\mu_i t_i = \mu_j t_j, \quad i \ne j, \tag{1}$$

where n_i and n_j are any pair of *sibling* nodes in a hierarchy.

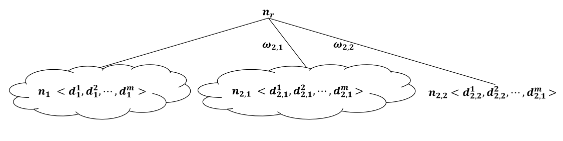
2) Low Complexity: To handle a large amount of jobs submitted to datacenters, the scheduler must operate with a low complexity, defined as the time required to make a task scheduling decision.

3) Constraints: Multi-resource allocation must satisfy the capacity constraints of all the m types of resources. Once a resource j is saturated, the allocation finishes. However, for a job that only requires k ($k \ne j$), it can continue to increase its allocation until k is fully occupied. For example, if a job requires $\langle 0 \text{ CPUs}, 1 \text{ GB}\rangle$ for each of its tasks, it continues to increase its allocation on the memory until all memory is used up, no matter what the consumption status of CPU is.



n_r
$n_1 < d_1^1, d_1^2, \cdots, d_1^m >$
$n_2 < c_1(s_{2,1}^1 + s_{2,2}^1), c_2(s_{2,1}^2 + s_{2,2}^2), \cdots, c_m(s_{2,1}^m + s_{2,2}^m) >$
$n_{2,1} < d_{2,1}^1, d_{2,1}^2, \cdots, d_{2,1}^m >$
$n_{2,2} < d_{2,2}^1, d_{2,2}^2, \cdots, d_{2,1}^m >$

(a) General hierarchy for the leaf node that requires every resource.



n_r
$\omega_{2,1}$
$\omega_{2,2}$
$n_1 < d_1^1, d_1^2, \cdots, d_1^m >$
$n_{2,1} < d_{2,1}^1, d_{2,1}^2, \cdots, d_{2,1}^m >$
$n_{2,2} < d_{2,2}^1, d_{2,2}^2, \cdots, d_{2,1}^m >$

(b) Transformation of Fig. 2a.

Fig. 2. Weight determination for the leaf node that requires every resource.

Therefore, in this letter, we discuss the allocation considering two types of jobs: *full-type* and *partial-type*. A job submitting tasks requiring every type of resource is referred to as a full-type job, while a job that does not require every resource is referred to as a partial-type job. Throughout this letter, we assume that a partial-type job has positive demands for only one of the resource types. This discussion can be easily generalized to the cases when jobs ask for two or more but not all resource types.

III. Multi-Resource Collapsed Hierarchies

A. Multi-Resource Collapsed Hierarchies (MCH)

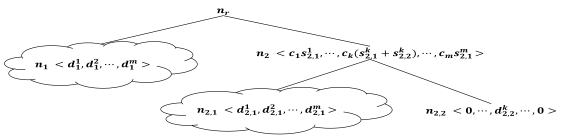
1) Overview: MCH converts a hierarchy into a flat one. The idea is to take the hierarchy specification and determine the corresponding weights for each leaf if the hierarchy is flattened. MCH then uses the flattened tree as the original weighted DRF scheduler to allocate resources. Therefore, we mainly focus on the weights determination in MCH.

2) Generating Demands of Internal Nodes: In order to compute the weights in the flattened tree, we should first define the resource demands of internal nodes. The key feature of internal nodes is to schedule resources in a way such that its children can increase their dominant shares at the same speed. Therefore, we generate the demands of an internal node using the *normalized demands* of its children. The normalized demand of user i on resource r is defined as $s_i^j = \frac{d_i^j}{c_j \mu_i}$. For example, for a leaf node n_i with resource demand $\langle 2 \text{ CPUs}, 3 \text{ GB}\rangle$ submitted to a resource pool with capacity $\langle 10 \text{ CPUs}, 12 \text{ GB}\rangle$, the resource share of each task from n_i is $\langle 2/10, 3/12\rangle = \langle 1/5, 1/4\rangle$, leading μ_i of n_i to be 1/4, then the normalized resource demand of n_i is $\langle (1/5)/(1/4), (1/4)/(1/4)\rangle$, that is, $\langle 4/5, 1\rangle$. Then the resource demand of an internal node $\tilde{n}$ on resource j can be denoted as follows,

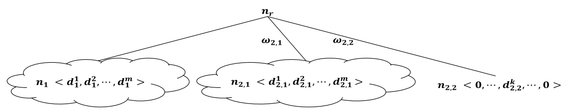
$$\tilde{d}^j = c_j \sum_{i \in C(\tilde{n})} s_i^j \tag{2}$$

3) Weights Computation in the Flattened Tree: We compute the weights of the jobs in two cases: full-type jobs and partial-type jobs. We denote the first saturated resource type in the datacenter as k', $k' \in \mathcal{R}$. Let t_i^* be the number of tasks allocated to job i in the hierarchical tree while t_i be the number of tasks allocated to job i in the flattened tree.

Case 1: Weights computation for full-type jobs. Consider the minimal general case shown in Fig. 2a, $n_{2,2}$ is a full-type job (*i.e.*, $d_{2,2}^j > 0, \forall j \in \mathcal{R}$). n_1 and $n_{2,1}$ can be regarded as


n_r
$n_1 < d_1^1, d_1^2, \cdots, d_1^m >$
$n_2 < c_1 s_{2,1}^1, \cdots, c_k(s_{2,1}^k + s_{2,2}^k), \cdots, c_m s_{2,1}^m >$
$n_{2,1} < d_{2,1}^1, d_{2,1}^2, \cdots, d_{2,1}^m >$
$n_{2,2} < 0, \cdots, d_{2,2}^k, \cdots, 0 >$

(a) General hierarchy for the leaf node that requires one type of res


n_r
$\omega_{2,1}$
$\omega_{2,2}$
$n_1 < d_1^1, d_1^2, \cdots, d_1^m >$
$n_{2,1} < d_{2,1}^1, d_{2,1}^2, \cdots, d_{2,1}^m >$
$n_{2,2} < 0, \cdots, d_{2,2}^k, \cdots, 0 >$

(b) Transformation of Fig. 3a.

Fig. 3. Weight computation for the leaf node that requires one type of resources.

virtual nodes. A virtual node can be regarded as a collection any structure of hierarchical nodes. The resource allocation to n_2 is always the summation of the resource allocations to $n_{2,1}$ and $n_{2,2}$. Therefore, from Fig. 2a we have

$$t_2^* \ c_j(s_{2,1}^j + s_{2,2}^j) = t_{2,1}^* d_{2,1}^j + t_{2,2}^* d_{2,2}^j, \quad \forall j \in \mathcal{R}. \tag{3}$$

According to the definition of normalized resource demand, we have

$$t_2^* = t_{2,1}^* \mu_{2,1} = t_{2,2}^* \mu_{2,2}. \tag{4}$$

Then from Fig. 2a we have

$$\mu_1 t_1^* = \mu_2 t_2^*; \mu_{2,1} t_{2,1}^* = \mu_{2,2} t_{2,2}^*, \tag{5a}$$

$$d_1^{k'} t_1^* + d_{2,1}^{k'} t_{2,1}^* + d_{2,2}^{k'} t_{2,2}^* = c_{k'}. \tag{5b}$$

(5a) is true because in the hierarchy n_1 and n_2, $n_{2,1}$ and $n_{2,1}$ are sibling nodes, respectively, and sibling nodes deserve a equalization in dominant shares. (5b) represents that resource k' is saturated. From Fig. 2b we have

$$\mu_1 t_1 = \frac{\mu_{2,1} t_{2,1}}{w_{2,1}} = \frac{\mu_{2,2} t_{2,2}}{w_{2,2}}, \tag{6a}$$

$$d_1^{k'} t_1 + d_{2,1}^{k'} t_{2,1} + d_{2,2}^{k'} t_{2,2} = c_{k'}. \tag{6b}$$

(6a) is true because in Fig. 2b n_1, $n_{2,1}$ and $n_{2,2}$ are sibling nodes, while (6b) is true because resource k' is fully occupied.

In order to transform the hierarchy from Fig. 2a to the flat tree from Fig. 2b, we hope that the allocation in Fig. 2b is exactly the same with the allocation in Fig. 2a, that is, $t_1 = t_1^*$, $t_{2,1} = t_{2,1}^*$, $t_{2,2} = t_{2,2}^*$. Therefore, the weights of $n_{2,1}$ and $n_{2,2}$ are $w_{2,1} = w_{2,2} = \frac{1}{\mu_2}$.

For a general hierarchy (an arbitrary structured tree where the weight on each node of the tree is 1), the weight of a full-type job n_i in the flattened tree can be summarized as $w_i = \prod_{h=1}^{H-1} \frac{1}{\mu_{P^h(i)}}$.

Case 2: Weights computation for partial-type jobs. Consider the minimal general case shown in Fig. 3, $n_{2,2}$ is a partial-type job that has positive demands for only one of the resource types k (*i.e.*, $d_{2,2}^k > 0$ and $d_{2,2}^j = 0, \forall j \in \mathcal{R}, j \neq k$), while n_1 and $n_{2,1}$ are still virtual nodes. Next we need to compute the weights in two sub-cases: $k = k'$ and $k \neq k'$.

Sub-case 1: When $k' \neq k$, from Fig. 3a we have

$$\mu_1 t_1^* = \mu_2 t_2^*, \tag{7a}$$

$$d_1^{k'} t_1^* + d_{2,1}^{k'} t_{2,1}^* = c_{k'}. \tag{7b}$$

Equation (7a) is true because n_1 and n_2 are a pair of siblings nodes and their dominant shares are the same until k' is saturated. Equation (7b) denotes that k' is saturated. Since $t_2^* = t_{2,1}^* \mu_{2,1}$ is always true, t_1^* and $t_{2,1}^*$ can be determined from Equations (7a) and (7b). At this point, $n_{2,2}$ continues to increase its dominant share as the resource it asks for is not saturated. When resource k is exhausted, we have

$$d_1^k t_1^* + d_{2,1}^k t_{2,1}^* + d_{2,2}^k t_{2,2}^* = c_k. \tag{8}$$

From Equation (8) we obtain the allocation to $n_{2,2}$, which is $t_{2,2}^*$. Next, from the hierarchy in Fig. 3b we have

$$\mu_1 t_1 = \frac{\mu_{2,1} t_{2,1}}{w_{2,1}}, \tag{9a}$$

$$d_1^{k'} t_1 + d_{2,1}^{k'} t_{2,1} = c_{k'}, \tag{9b}$$

$$d_1^k t_1 + d_{2,1}^k t_{2,1} + d_{2,2}^k t_{2,2} = c_k. \tag{9c}$$

Equation (9a) denotes that the weighted dominant shares of n_1 and $n_{2,1}$ in the flat tree are the same, and (9b) and (9c) represents that the allocation terminates when both k and k' are saturated.

Still, we hope that $t_1 = t_1^*$, $t_{2,1} = t_{2,1}^*$, $t_{2,2} = t_{2,2}^*$. (7), (8) and (9) together yield $w_{2,1} = \frac{1}{\mu_2}$, while $w_{2,2}$ remaining to be the same, that is $w_{2,2} = 1$.

For a general hierarchy, the weight of a partial-type job n_i that requires only resource k, which is not the system's first saturated resource type k', in the flattened tree can be summarized as $w_i = 1$.

Sub-case 2: When $k' = k$, from Fig. 3a we have

$$\mu_1 t_1^* = \mu_2 t_2^*, \tag{10a}$$

$$\mu_{2,1} t_{2,1}^* = \mu_{2,2} t_{2,2}^*, \tag{10b}$$

$$d_1^k t_1 + d_{2,1}^k t_{2,1} + d_{2,2}^k t_{2,2} = c_k. \tag{10c}$$

In this case, the dominant shares of $n_{2,1}$ and $n_{2,2}$ will be strictly equalized (see Equation (10b)) as there is no more spare resource k to be further allocated to $n_{2,2}$. Equation (10c) indicates that k is fully exhausted.

Next, from the flat tree shown in Fig. 3b we have

$$\mu_1 t_1 = \frac{\mu_{2,1} t_{2,1}}{w_{2,1}} = \frac{\mu_{2,2} t_{2,2}}{w_{2,2}}, \tag{11a}$$

$$d_1^k t_1 + d_{2,1}^k t_{2,1} + d_{2,2}^k t_{2,2} = c_k. \tag{11b}$$

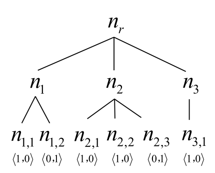
(11a) holds as resource k is the first saturated resource and the weighted dominant shares of n_1, $n_{2,1}$ and $n_{2,2}$ can always be equalized since no more resource k can be further allocated to $n_{2,2}$. Still, we hope that $t_1 = t_1^*$, $t_{2,1} = t_{2,1}^*$, $t_{2,2} = t_{2,2}^*$. To achieve this objective, we obtain $w_{2,1} = w_{2,2} = \frac{1}{\mu_2}$.

For a general hierarchy, the weight of n_i that requires only resource k, which is exactly the system's first saturated resource k', in the flattened tree can be summarized as $w_i = \prod_{h=1}^{H-1} \frac{1}{\mu_{P^h(i)}}$.

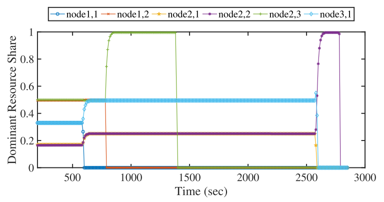
B. Time Complexity of MCH

Theorem 1: The run time of MCH is $O(q)$, where q is the number of all nodes in the hierarchy.

Proof: MCH consists of two parts. The first part is the flattening step that transforms the hierarchy into a weighted flat tree. This process requires a traversal of all the nodes in



(a) Hierarchy used in Section IV



(b) Dynamic allocation of the leaves shown in Fig. 4a

Fig. 4. Effectiveness of MCH.

the hierarchy with a run time of $O(q)$. The second part is for the flattened tree to implement weighted DRF. Thus its run time is the same with the run time of the weighted DRF, which is $O(\log p)$. Therefore, the time complexity of MCH is $O(q + \log p) = O(q)$. ■

IV. Simulation

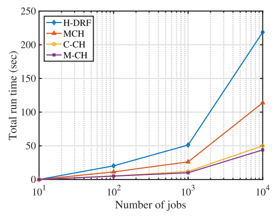
A. Hierarchical Share Guarantee

We first confirm experimentally the effectiveness of MCH in providing hierarchical share guarantee. We have implemented MCH using a simple, easy-to-verify hierarchy shown in Fig. 4a. Each of the 6 jobs submits 2000 tasks and the overall capacity of the cluster is $\langle 200 \text{ CPUs}, 200 \text{ GB memory}\rangle$. Fig. 4b shows the dominant shares allocated to the leaves across time. We observe that when all the leaves the active (between 200-600s), the dominant share of the leaves are 0.33, 0.5, 0.165, 0165, 0.5, 0.33 from left to right. This is indeed the case. After flattening the hierarchy, the weights of all leaf nodes would be 1, 1, 0.5, 0.5, 1, 1 from left to right. The 2:1:1:2 ratio of weights among $n_{1,1}$, $n_{2,1}$, $n_{2,2}$ and $n_{3,1}$ is exactly consistent with the ratio of dominant shares of the corresponding nodes (*i.e.*, 0.33:0.165:0.165:0.33 = 2:1:1:2). Meanwhile, as $n_{1,2}$ and $n_{2,3}$ only ask for memory, they can continue to increase their dominant share and get an equally split of memory without limitation to the dominant shares of their sibling nodes. This property is also exhibited every time a node leaves the hierarchy (*e.g.*, when $n_{1,2}$ leaves at 790s, the dominant share of $n_{2,3}$ increases to 1 while the dominant shares of others remain to be the same).

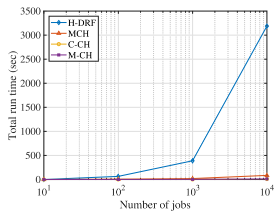
B. Run Time Comparison

We compare the total run time of MCH with H-DRF, CH w.r.t CPU share (hereafter C-CH) and CH w.r.t Mem share (hereafter M-CH). We use full binary trees with 2 to 5 levels as experimental hierarchies. With each hierarchy, we evaluate the total run time in scheduling different number of jobs sampled from Google cluster-data traces [10].

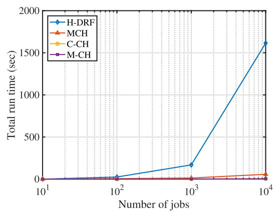
From Fig. 5 we observe that for hierarchies with 3 to 5 levels, the total run time of H-DRF is almost the square of MCH, just as the theory indicates, saving a great deal of computation time. However, for the simplest 2-level hierarchy, MCH saves only half of the total run time (see Fig. 5a), which is not align with the improvement from $O(q^2)$ to $O(q)$ (here $q = 6$). We attribute this phenomenon to the fact that a two-level full binary hierarchy has a simple structure with only 4 leaves, the scheduling of 4 leaves are so easy that the advantage of MCH is not obvious in saving the run time. Meanwhile, MCH behaves no better than C-CH or M-CH. Both C-CH and M-CH need simple weights' computation in flattening the hierarchy regardless of the jobs' resource demands. Their time complexities are the same with DRF, which is $O(\log p)$, much smaller than the complexity of MCH.



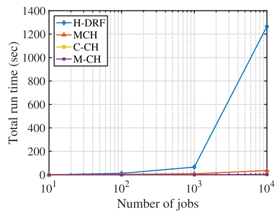
(a) 2-level full binary tree



(b) 3-level full binary tree



(c) 4-level full binary tree



(d) 5-level full binary tree

Fig. 5. Total run time comparison for different number of jobs.

V. Conclusions

We proposed an online multi-resource allocation algorithm, MCH, that supported both hierarchies and fairness with a linear time complexity $O(q)$. MCH converted the hierarchical tree in to a weighted flat one and then scheduled resources as a flat scheduler. MCH provided the hierarchical share guarantees by separating jobs into two categories: full-type jobs and partial-type jobs, and then MCH computed the weights for the jobs in each category if the hierarchy was flattened. We had also validated out theoretical results via extensive simulation studies.

References

[1] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, “Dominant resource fairness: Fair allocation of multiple resource types,” in *Proc. NSDI*, 2011, p. 24.
[2] J. Khamse-Ashari, I. Lambadaris, G. Kesidis, B. Urgaonkar, and Y. Zhao, “An efficient and fair multi-resource allocation mechanism for heterogeneous servers,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 12, pp. 2686–2699, Dec. 2018.
[3] A. A. Bhattacharya, D. Culler, E. Friedman, A. Ghodsi, S. Shenker, and I. Stoica, “Hierarchical scheduling for diverse datacenter workloads,” in *Proc. SoCC*, 2013, pp. 1–4.
[4] W. Wang, B. Liang, and B. Li, “Multi-resource fair allocation in heterogeneous cloud computing systems,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 10, pp. 2822–2835, Oct. 2015.
[5] W. Wang, B. Li, B. Liang, and J. Li, “Multi-resource fair sharing for datacenter jobs with placement constraints,” in *Proc. SC*, Nov. 2016, pp. 1003–1014.
[6] D. C. Parkes, A. D. Procaccia, and N. Shah, “Beyond dominant resource fairness: Extensions, limitations, and indivisibilities,” *ACM Trans. Econ. Comput.*, vol. 3, no. 1, p. 3, 2015.
[7] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, “Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling,” in *Proc. EuroSys*, 2010, pp. 265–278.
[8] B. Hindman *et al.*, “Mesos: A platform for fine-grained resource sharing in the data center,” in *Proc. NSDI*, 2011, p. 22.
[9] A. Chandra and P. Shenoy, “Hierarchical scheduling for symmetric multiprocessors,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 3, pp. 418–431, Mar. 2008.
[10] C. Reiss, J. Wilkes, and J. L. Hellerstein, “Google cluster-usage traces: Format + schema,” Google, Mountain View, CA, USA, White Paper, 2011, pp. 1–14.