Fairness-Efficiency Scheduling for Cloud Computing with Soft Fairness Guarantees

Shanjiang Tang, Ce Yu, Yusen Li

Abstract—Fairness and efficiency are two important metrics for users in modern data center computing system. Due to the heterogeneous resource demands of CPU, memory, and network I/O for users’ tasks, it cannot achieve the strict 100% fairness and the maximum efficiency at the same time. Existing fairness-efficiency schedulers (e.g., Tetris) can balance such a tradeoff elastically by relaxing fairness constraint for improved efficiency using the knob. However, their approaches are unaware of fairness degradation under different knob configurations, which makes several drawbacks. First, it cannot tell how much relaxed fairness can be guaranteed given a knob value. Second, it fails to meet several essential properties such as sharing incentive. To address these issues, we propose a new fairness-efficiency scheduler, QKnober, to balance the fairness and efficiency elastically and flexibly using a tunable fairness knob. QKnober is a fairness-sensitive scheduler that can maximize the system efficiency while guaranteeing the \(\theta\)-soft fairness by modeling the whole allocation as a combination of fairness-oriented allocation and efficiency-oriented allocation. Moreover, QKnober satisfies fairness properties of sharing incentive, envy-freeness and pareto efficiency given a proper knob value. We have implemented QKnober in YARN and evaluated it using both testbed and simulated experiments. The results show that QKnober can achieve good performance and fairness.

Index Terms—Multi-Resource Allocation, Fairness, Efficiency, Hadoop

---

**APPENDIX A**

**PROOF OF THEOREM 1**

Proof: For any two users \(i, j \in [1, n]\), we have

\[
\frac{s_i}{w_i} - \frac{s_j}{w_j} \leq \max_{1 \leq i, j \leq n} \left\{ \frac{s_i}{w_i} - \frac{s_j}{w_j} \right\} \\
= \max_{1 \leq i, j \leq n} \left\{ \frac{s_{i, j}^{\max} \cdot \rho + s_i - s_j}{w_i} \cdot \frac{s_i - s_j}{w_j} \right\} \\
= \max_{1 \leq i, j \leq n} \left\{ \frac{s_i - s_j}{w_i} \cdot \min_{1 \leq j \leq n} \frac{s_i - s_j}{w_j} \right\} \\
= \max_{1 \leq i, j \leq n} \left\{ \frac{U_i}{D_j}, \max_{1 \leq k \leq m} \left\{ \frac{d_{i, k}}{r_k} \right\} \right\} - \frac{U_j}{D_j}, \max_{1 \leq k \leq m} \left\{ \frac{d_{j, k}}{r_k} \right\} \right\}.
\]

Case #1: When \(\rho = 1\), we have \(s_i' = 0\) for all \(i \in [1, n]\) discussed in Section 4.2.1 of the main file. In that case, it holds \(\frac{s_i}{w_i} - \frac{s_j}{w_j} \leq 0\) for any user \(i, j \in [1, n]\).

Case #2: When \(0 \leq \rho < 1\), according to the soft fairness definition, our proof turns to be seeking for an upper bound \(\theta\) such that \(\max_{1 \leq i, j \leq n} \left\{ \frac{U_i'}{D_j}, \max_{1 \leq k \leq m} \left\{ \frac{d_{i, k}}{r_k} \right\} \right\} - \frac{U_j'}{D_j}, \max_{1 \leq k \leq m} \left\{ \frac{d_{j, k}}{r_k} \right\} \right\} \leq \theta\) for all feasible allocations under the resource capacity vector \(R\).

In the efficiency-oriented resource allocation, a feasible allocation \(U' = \langle U'_1, \ldots, U'_n \rangle\) ceases when at least one resource is fulfilled, i.e.,

\[
\max_{1 \leq i \leq m} \left\{ \frac{U'_i}{D_i}, \max_{1 \leq k \leq m} \left\{ \frac{d_{i, k}}{r_k} \right\} \right\} = 1.
\]

Additionally, for all feasible allocations, the maximum value of \(U'_j\) exists for user \(i\) when it possesses all the resource capacity vector \(R\) exclusively. In that case, all other users have no resource allocations, i.e., \(\forall j \neq i \in [1, n], U'_j = 0\). We then get the maximum value of \(U'_i\) according to Formula (5) and (19) as follows:

\[
U'_i = \frac{D_i}{\max_{1 \leq k \leq m} \left\{ \frac{d_{i, k}}{r_k} \right\}}.
\]

The upper bound \(\theta\) of Formula (18) for all feasible allocations can then be computed, i.e.,

\[
\theta = \max_{1 \leq i \leq n} \left\{ \frac{U'_i}{D_i}, \max_{1 \leq k \leq m} \left\{ \frac{d_{i, k}}{r_k} \right\} \right\} - \frac{U_j'}{D_j}, \max_{1 \leq k \leq m} \left\{ \frac{d_{j, k}}{r_k} \right\} \right\}.
\]

**APPENDIX B**

**PROOF OF THEOREM 2**

Proof: Let’s start with the exclusively non-sharing case, where each user \(i\) schedules tasks under its own partition of the system resource, i.e., \(\sum_{j=1}^{w_i} u_j = \sum_{j=1}^{w_i} u_j = R\). In this case, the allocation stops when at least one resource is saturated, i.e., \(\max_{1 \leq k \leq m} \left\{ \frac{d_{i, k}}{r_k} \right\} = N_i(U_i)/(r_k \cdot \sum_{j=1}^{w_i} u_j)\). We then get the maximum value of \(U_i\) for user \(i\), i.e.,

\[
N_i(U_i) = 1/ \max_{1 \leq k \leq m} \left\{ \frac{d_{i, k}}{r_k} \cdot \frac{u_i}{\sum_{j=1}^{w_i} u_j} \right\}.
\]
Now we consider the sharing case for each user \( i \). According to Formula (8), (14) and (11), we have
\[
N_i(U_i) = \frac{s_i}{\max_{1 \leq k \leq m} \{d_{i,k}/r_k\}} = \frac{s_i^{\max} + s'_i}{\max_{1 \leq k \leq m} \{d_{i,k}/r_k\}} = \frac{w_i}{\max_{1 \leq k \leq m} \{d_{i,k}/r_k\}} \geq N_i(U_1).
\]

**APPENDIX C**

**PROOF OF THEOREM 3**

**Proof:** By contradiction, let's assume that user \( i \) envies the allocation result of user \( j \) under QKnobler allocation policy. Then for user \( i \), it must have
\[
N_i(U_i) < N_i(U_j). \tag{21}
\]
We consider the following two cases:

**Case 1:** \( \frac{D_i}{D_j} = \frac{D_j}{D_i} \): according to the fairness requirement of Formula (15), Formula (14) and (8), we have
\[
\frac{w_i}{w_j} = \frac{s_i}{s_j} \Leftrightarrow \frac{N_i(U_i)}{N_j(U_j)} = \frac{d_{i,k}/r_k}{d_{j,k}/r_k} = \frac{N_i(U_i) \cdot \max_{1 \leq k \leq m} \{d_{i,k}/r_k\}}{N_j(U_j) \cdot \max_{1 \leq k \leq m} \{d_{j,k}/r_k\}} \cdot \frac{d_{j,k}/r_k}{d_{i,k}/r_k}.
\]
By exchanging the allocation between user \( i,j \), subject to the fairness constraint of Formula (15), we should have
\[
\frac{N_i(U_i)}{N_j(U_j)} = \frac{d_{i,k}/r_k}{d_{j,k}/r_k} = \frac{N_j(U_j)}{N_i(U_i)} \cdot \max_{1 \leq k \leq m} \{d_{j,k}/r_k\} \cdot \max_{1 \leq k \leq m} \{d_{i,k}/r_k\} \cdot \frac{d_{j,k}/r_k}{d_{i,k}/r_k}.
\]
Hence, we have \( N_i(U_i)/N_j(U_j) = N_j(U_j)/N_i(U_i) \).\tag{22}

Moreover, \( \frac{D_i}{D_j} = \frac{D_j}{D_i} \Rightarrow \frac{d_{i,k}}{d_{j,k}} = \frac{D_i}{D_j} = 1 \), \( \forall k \in [1,m] \). According to Formula (4), we have
\[
N_i(U_j) = \min_{1 \leq k \leq m} \{u_{j,k}/d_{i,k}\} = N_j(U_j) \cdot \min_{1 \leq k \leq m} \{d_{j,k}/d_{i,k}\} = N_j(U_j) \cdot [D_j]/[D_i].\tag{23}
\]
Similarly, we have
\[
N_j(U_i)/N_i(U_i) = [D_i]/[D_j]. \tag{24}
\]
According to Formula (22), (23) and (24), we have
\[
N_i(U_i)/N_j(U_j) = N_j(U_j)/N_i(U_i) = N_j(U_j) = N_i(U_j), \tag{25}
\]
which contradicts the assumption of Formula (21).

**Case 2:** \( \frac{D_i}{D_j} \neq \frac{D_j}{D_i} \): According to Formula (4), we have
\[
N_i(U_j) = \min_{1 \leq k \leq m} \{u_{j,k}/d_{i,k}\} = N_j(U_j) \cdot \min_{1 \leq k \leq m} \{d_{j,k}/d_{i,k}\}. \tag{25}
\]
and
\[
N_j(U_i) = \min_{1 \leq k \leq m} \{u_{i,k}/d_{j,k}\} = N_i(U_j) \cdot \min_{1 \leq k \leq m} \{d_{i,k}/d_{j,k}\}. \tag{26}
\]
According to Formula (6), there are
\[
\epsilon_i(U_j) = N_i(U_j) \cdot \sum_{k=1}^{m} d_{i,k}/r_k \leq N_j(U_j) \cdot \min_{1 \leq k \leq m} \{d_{j,k}/d_{i,k}\} \cdot \sum_{k=1}^{m} d_{i,k}/r_k = \epsilon_j(U_j) \cdot \min_{1 \leq k \leq m} \{d_{j,k}/d_{i,k}\} \cdot \sum_{k=1}^{m} d_{i,k}/r_k. \tag{27}
\]

Furthermore, since \( \frac{D_i}{[D_j]} \neq \frac{D_j}{[D_i]} \), the following two conditions must hold:
\[
i). \min_{1 \leq k \leq m} \{d_{j,k}/r_k\} \leq d_{i,k}/r_k \Rightarrow d_{j,k}/r_k \geq d_{i,k}/r_k \cdot \min_{1 \leq k \leq m} \{d_{j,k}/r_k\}, \forall k \in [1,m]. \tag{28}
\]
\[
ii). \min_{1 \leq k \leq m} \{d_{j,k}/r_k\} < d_{j,k}/r_k \Rightarrow d_{j,k}/r_k > d_{i,k}/r_k \cdot \min_{1 \leq k \leq m} \{d_{j,k}/r_k\}, \exists k_1 \in [1,m]. \tag{29}
\]
According to Formula (28) and (29), we have
\[
\min_{1 \leq k \leq m} \{d_{j,k}/r_k\} \cdot \sum_{k=1}^{m} d_{j,k}/r_k < \sum_{k=1}^{m} d_{i,k}/r_k \cdot \sum_{k=1}^{m} d_{j,k}/r_k \cdot \sum_{k=1}^{m} d_{i,k}/r_k = 1.
\]
Hence, we have \( \epsilon_i(U_j) < \epsilon_j(U_j) \) according to Formula (27). Similarly, it follows that \( \epsilon_j(U_i) < \epsilon_i(U_i) \). Then after swapping the resource allocation, we have \( \epsilon_i(U_j) + \epsilon_j(U_i) < \epsilon_i(U_j) + \epsilon_j(U_j) \), which violates the efficiency maximization requirement in Section 4.1 of the main file and hereby the assumption is false.

Finally, according to Case 1 and 2, we now can safely make the conclusion that QKnobler policy is envy-freeness. \( \square \)

**APPENDIX D**

**PROOF OF THEOREM 4**

**Proof:** We prove this theorem by contradiction to suppose that the resulting allocation \( U = (U_1, \cdots, U_n) \) of QKnobler is not Pareto efficient. Then there must exist an alternative allocation \( U = (U_1, \cdots, U_n) \) such that \( N_i(U_i) \leq N_i(U_j) \) for \( \forall i \in [1,m] \) and \( \exists j \in [1,m], N_i(U_j) < N_j(U_j) \). Similar to DRF mentioned in Section 4.1 of the main file, QKnobler follows the progressive filling and the allocation terminates when at least one resource is saturated. It means that for the allocation \( U \) of QKnobler, it holds
\[
\max_{1 \leq k \leq m} \{\sum_{i=1}^{n} u_{i,k}/r_k\} = \max_{1 \leq k \leq m} \{\sum_{i=1}^{n} N_i(U_i) \cdot d_{i,k}/r_k\} = 1. \tag{24}
\]
Thus,
\[
\max_{1 \leq k \leq m} \{\sum_{i=1}^{n} N_i(U_i) \cdot d_{i,k}/r_k\} > \max_{1 \leq k \leq m} \{\sum_{i=1}^{n} N_i(U_i) \cdot d_{i,k}/r_k\} = 1.
\]
which is not a feasible allocation and indicates that the assumption does not hold. Therefore, QKnobler is Pareto efficient. \( \square \)

**APPENDIX E**

**Discrete Resource Allocation**

In the previous section, we have implicitly assumed one ‘supercomputer’ with all big resources that can be allocated in arbitrarily small units. However, in practice, it is more likely to have a data center cluster consisting of many small computing nodes, which are allocated to tasks in discrete amounts. We refer to these two scenarios as the continuous, and the discrete scenario,
respectively. We now take a look at how fairness is affected in the discrete scenario.

Consider a cluster consisting of \( K \) computing nodes, where the resource capacity of the \( i \)th machine is \( \mathcal{R}_i = \langle r_{i,1}, \cdots, r_{i,m} \rangle \) and \( \mathcal{R} = \sum_{i \in K} \mathcal{R}_i \). We assume that any task can be scheduled on every computing node. We further assume that each user has strictly positive demands. With these assumptions, we have the following conclusion,

**Theorem 1:** In the discrete scenario, QKNober is a \( \theta \)-soft fairness policy where the difference between the allocations of any two users is bounded by

\[
\theta = \begin{cases} 
\max_{1 \leq i \leq n} \frac{\max_{1 \leq k \leq m} \frac{d_{i,k}}{w_i}}{\max_{1 \leq i \leq n} \frac{d_{i,k}}{w_i}} \left( \frac{\max_{1 \leq k \leq m} \frac{d_{i,k}}{r_k}}{\max_{1 \leq k \leq m} \frac{d_{i,k}}{r_k}} \right), & (0 \leq \rho < 1), \\
\max_{1 \leq i \leq n} \left\{ \max_{1 \leq i' < n} \frac{\max_{1 \leq k \leq m} \frac{d_{i,k}}{w_i}}{\max_{1 \leq k \leq m} \frac{d_{i,k}}{w_i}} \right\} & (\rho = 1) 
\end{cases}
\]

**Proof:** In Section 4.2.1 of the main file, we have theoretically shown that QKNober can guarantee \( \bar{s}_i = s_i^{\text{max}} \cdot \rho \) soft fairness in the fairness-oriented allocation under the continuous scenario by assuming the number of tasks can be partial value and one supercomputer containing all computing resources. However, in the discrete scenario, both task number and machine number are discrete integer, indicating that it is hard or even impossible to achieve the exact value of \( s_i^{\text{max}} \cdot \rho \) soft fairness in its fairness-oriented allocation. Instead, we can seek to guarantee a soft fairness \( \bar{s}_i \) in the fairness-oriented allocation under the discrete scenario satisfying that

\[
s_i^{\text{max}} \cdot \rho \leq \bar{s}_i \leq s_i^{\text{max}} \cdot \rho + \max_{1 \leq i' \leq n} \left\{ \max_{1 \leq k \leq m} \frac{d_{i',k}}{r_k} \right\}.
\]

for each user \( i \). Hence it holds,

\[
0 \leq \left| \frac{s_i}{w_i} - \frac{s_j}{w_j} \right| \leq \max_{1 \leq i' \leq n} \frac{\max_{1 \leq k \leq m} d_{i',k}/r_k}{w_{i'}}.
\]

**Case #1:** When \( \rho = 1 \), the system performs pure fairness allocation only for the whole cluster resources. In that case, \( s_i = 0 \) for \( \forall i \in [1, n] \). Then

\[
\left| \frac{s_i}{w_i} - \frac{s_j}{w_j} \right| \leq \max_{1 \leq i, j \leq n} \left\{ \left| \frac{s_i}{w_i} - \frac{s_j}{w_j} \right| \right\} \\
\leq \max_{1 \leq i, j \leq n} \left\{ \left( \frac{s_i}{w_i} - \frac{s_j}{w_j} \right) \right\} + \left( \frac{s_i}{w_i} - \frac{s_j}{w_j} \right) \\
\leq \max_{1 \leq i, j \leq n} \left\{ \left( \frac{s_i}{w_i} - \frac{s_j}{w_j} \right) \right\}
\]

Case #2: When \( 0 \leq \rho < 1 \), for any two users \( i, j \in [1, n] \) that

\[
\left| \frac{s_i}{w_i} - \frac{s_j}{w_j} \right| \leq \max_{1 \leq i, j \leq n} \left\{ \left| \frac{s_i}{w_i} - \frac{s_j}{w_j} \right| \right\} \\
\leq \max_{1 \leq i, j \leq n} \left\{ \left( \frac{s_i}{w_i} - \frac{s_j}{w_j} \right) \right\}
\]

**APPENDIX F**

**OVERHEAD EVALUATION**

Recall from our system implementation in Section 6.2 that the task scheduling logic (e.g., fairness-oriented allocation plus
efficiency-oriented allocation) of QKnober is more complicated than that in YARN. This section evaluates the computational overhead of QKnober with the aforementioned four workloads under different knob configurations. Specifically, we consider the time taken by the Resource Manager (RM) to deal with a heartbeat request from the Node Manager (NM) and from the Application Master (AM). In YARN, the RM performs the actual resource allocation during the NM heartbeat. AM is responsible for asking RM for resource allocation. At an AM ask heartbeat, the RM updates the accumulative asks from the AM and reacts with any tasks in the past asks that have been satisfied in the NM heartbeat.

Figure 1 presents the overhead results for DRF and QKnober to process heartbeats from the NM and AM under different knob values, respectively. It shows that the heartbeats processing time for both NM and AM is minor compared with the Hadoop workloads that often takes hours or days to complete [32]. Second, QKnober performs heartbeats a bit slower than DRF. This is because QKnober has more complex task scheduling mechanism than DRF. Third, for QKnober, the heartbeats processing time is much close under different knob configurations. It indicates that the knob configuration has no too much impact on the overhead contribution.

ACKNOWLEDGEMENT
This work is sponsored by the National Natural Science Foundation of China (61972277) and Tianjin Natural Science Foundation (18JCZDJC30800). Ce Yu is supported by the Joint Research Fund in Astronomy (U1731243, U1931130) under cooperative agreement between the National Natural Science Foundation of China (NSFC) and Chinese Academy of Sciences (CAS).

REFERENCES

[34] Wei Wang, Chen Feng, Baochun Li, and Ben Liang. On the fairness-efficiency tradeoff for packet processing with multiple resources. In CoNEXT '14, pages 235–248, New York, NY, USA, 2014. ACM.


[37] Wei Wang, Shiyou Ma, Bo Li, and Baochun Li. Coflex: Navigating the fairness-efficiency tradeoff for coflow scheduling. In INFOCOM’17.

