Blind GB-PANDAS: A Blind Throughput-Optimal Load Balancing Algorithm for Affinity Scheduling

Ali Yekkehkhany and Rakesh Nagi

***Abstract*— Dynamic affinity load balancing of multi-type tasks on multi-skilled servers, when the service rate of each task type on each of the servers is known and can possibly be different from each other, is an open problem for over three decades. The goal is to do task assignment on servers in a real time manner so that the system becomes stable, which means that the queue lengths do not diverge to infinity in steady state (throughput optimality), and the mean task completion time is minimized (delay optimality). The fluid model planning, Max-Weight, and c-μ-rule algorithms have theoretical guarantees on optimality in some aspects for the affinity problem, but they consider a complicated queueing structure and either require the task arrival rates, the service rates of tasks on servers, or both. In many cases that are discussed in the introduction section, both task arrival rates and service rates of different task types on different servers are unknown. In this work, the Blind GB-PANDAS algorithm is proposed which is completely blind to task arrival rates and service rates. Blind GB-PANDAS uses an exploration-exploitation approach for load balancing. We prove that Blind GB-PANDAS is throughput optimal under arbitrary and unknown distributions for service times of different task types on different servers and unknown task arrival rates. Blind GB-PANDAS desires to route an incoming task to the server with the minimum weighted-workload, but since the service rates are unknown, such routing of incoming tasks is not guaranteed which makes the throughput optimality analysis more complicated than the case where service rates are known. Our extensive experimental results reveal that Blind GB-PANDAS significantly outperforms existing methods in terms of mean task completion time at high loads.**

***Index Terms*— Affinity scheduling, exploration-exploitation, near-data scheduling, data locality, data center, big data.**

I. Introduction

Affinity load balancing refers to allocation of computing tasks on computing nodes in an efficient way to minimize a cost function, for example the mean task completion time [1]. Due to the fact that different task types can have different processing (service) rates on different computing nodes (servers), a dilemma between throughput and delay optimality emerges which makes the optimal affinity load balancing an open problem for more than three decades if the task arrival rates are unknown. If the task arrival rates and the service rates of different task types on different servers are known, the fluid model planning algorithm by Harrison [2] and Harrison and Lopez [3], and Bell *et al.* [4], [5] is a delay optimal load balancing algorithm that solves a linear programming optimization problem to determine task assignment on servers. The same number of queues as the number of task types is needed for the fluid model planning algorithm, so the queueing structure is fixed to the number of task types and does not capture the complexity of the system model, which is how heterogeneous the service rates of task types on different servers are. As an example given in [6] and [7], for data centers with a rack structure that use Hadoop for map-reduce data placement with three replicas of data chunks on the M severs, the fluid model planning algorithm requires $\binom{M}{3}$ queues, while Xie *et al.* [6] propose a delay optimal algorithm that uses $3M$ queues. As another extreme example, if the service rates of N_T number of task types on all servers are the same, the fluid model planning algorithm still considers N_T number of queues, while the First-Come-First-Served (FCFS) algorithm uses a single queue and is both throughput and delay optimal. It is true that in the last example all task types can be considered the same type, but this is just an example to enlighten the reasoning behind the queueing structure for GB-PANDAS (Generalized Balanced Priority Algorithm for Near Data Scheduling) presented in Section II-A.

In the absence of knowledge on task arrival rates, Max-Weight [8] and c-μ-rule [9] algorithms can stabilize the system by just knowing the service rates of task types on different servers. None of these two algorithms are delay optimal though. The c-μ-rule is actually cost optimum, where it assumes convex delay costs associated to each task type, and minimizes the total cost incurred to the system. Since the cost functions have to be strictly convex, so cannot be linear, c-μ-rule does not minimize the mean task completion time. Since these two algorithms do not use the task arrival rates and still stabilize the system, they are robust to any changes in task arrival rate as long as it is in the capacity region of the system. Both Max-Weight and c-μ-rule algorithms have the same issue as the fluid model planning algorithm on considering one queue per task type which can make the system model complicated as discussed in [6]. Note that Wang *et al.* [10] and Xie *et al.* [6] study the load balancing problem for special cases of two and three levels of data locality, respectively. In the former, delay optimality is analyzed for a special traffic scenario and in the latter delay optimality is analyzed for a general traffic scenario and in both cases there is no issue on the number of queues, but as mentioned, these two algorithms are for special cases of two and three levels of data locality. Hence, a unified algorithm that captures the trade-off between the complexity of the queueing structure and the complexity of the system model is missed in the literature. Yekkehkhany *et al.* [11] implicitly mention this trade-off in

Manuscript received December 22, 2018; revised July 17, 2019 and November 6, 2019; accepted February 27, 2020; approved by IEEE/ACM Transactions on Networking Editor A. Eryilmaz. Date of publication March 19, 2020; date of current version June 18, 2020. *(Corresponding author: Ali Yekkehkhany.)*

Ali Yekkehkhany is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana–Champaign, Champaign, IL 61801-3028 USA (e-mail: yekkehk2@illinois.edu).

Rakesh Nagi is with the Department of Industrial and Enterprise Systems Engineering, University of Illinois at Urbana–Champaign, Champaign, IL 61801-3028 USA.

Digital Object Identifier 10.1109/TNET.2020.2978195

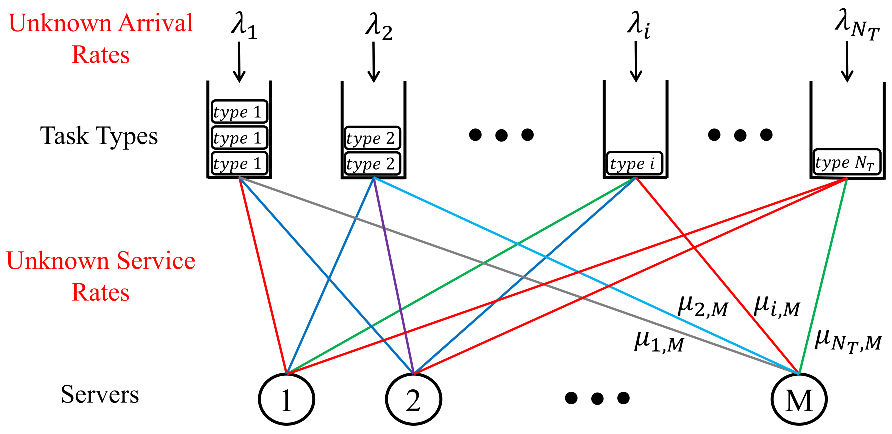

Fig. 1. Affinity scheduling setup with multi-type tasks and multi-skilled servers.

data center applications, but the generalization is not crystal clear and needs more thinking for the affinity setup, which is summarized in this work as a complementary note on the Balanced-PANDAS algorithm.

The affinity scheduling problem appears in different applications from data centers and modern processing networks that consist of heterogeneous servers, where data-intensive analytics like MapReduce, Hadoop, and Dryad are performed, to supermarket models, or even patient assignment to surgeons in big and busy hospitals and many more. Lack of dependable estimates of system parameters, including task arrival rates and specially service rates of task types on different servers, is a major challenge in constructing an optimal load balancing algorithm for such networks [12]. All the algorithms mentioned above at least require the knowledge of service rates of task types on different servers. In the absence of prior knowledge on service rates, such algorithms can be fragile and perform poorly, resulting in huge waste of resources. To address this issue, we propose a robust policy called Blind GB-PANDAS that is totally blind to all system parameters, but is robust to task arrival rate changes, learns the service rates of task types on different servers, so it is robust to any service rate parameter changes as well. It is natural that due to traffic load changes in data centers, the service rate of tasks on remote servers change over time. In such cases, Blind GB-PANDAS is capable of updating system parameters and taking action correspondingly. Blind GB-PANDAS uses an exploration-exploitation approach to make the system stable without any knowledge about the task arrival rates and the processing rates. More specifically, it uses an exploration-exploitation method, where in the exploration phase it takes action in a way to make the system parameter estimations more accurate, and in the exploitation phase it uses the estimated parameters to do an optimal load balancing based on the estimates. Note that only the processing rates of task types on different servers are the parameters that are estimated, and the task arrival rates are not estimated. The reason is that task arrival rates change frequently, so there is not a point on estimating them, whereas the service rates do not change rapidly. Since Blind GB-PANDAS uses an estimate of the processing rates, an incoming task is not necessarily routed to the server with the minimum weighted-workload in the exploitation phase, which increases complexity in the throughput optimality proof of Blind GB-PANDAS using the Lyaponuv-based method. The throughput optimality result is proved under arbitrary and unknown service time distributions with bounded means and bounded supports that do not necessarily require the memoryless property.

As discussed in Section II-A, the queueing structure used for Blind GB-PANDAS shows the trade-off between the heterogeneity of the underlying system model for processing rates and the complexity of the Blind GB-PANDAS queueing structure. Blind GB-PANDAS can also use a one queue per server queueing structure, where the workload on servers is of interest instead of the queue lengths, but for an easier explanation of the Blind GB-PANDAS algorithm we use multiple symbolic sub-queues for each server. The Blind GB-PANDAS algorithm is compared to FCFS, Max-Weight, and c-μ-rule algorithms in terms of average task completion time through simulations, where the same exploration-exploitation approach as Blind GB-PANDAS is used for Max-Weight and c-μ-rule. Our extensive simulations show that the Blind GB-PANDAS algorithm outperforms the three other algorithms at high loads by a reasonably large difference.

The rest of the paper is structured as follows. Section II describes the system model, GB-PANDAS, and the queueing structure of GB-PANDAS, in addition to deriving the capacity region of the system. Section III presents the Blind GB-PANDAS algorithm and queueing dynamics for this algorithm. Section IV starts with some preliminary results and lemmas and ends up with the throughput optimality proof for Blind GB-PANDAS. Section V evaluates the performance of Blind GB-PANDAS versus Max-Weight, c-μ-rule, and FCFS algorithms in terms of mean task completion time. Section VI discusses the related works, and finally Section VII concludes the paper with a discussion on opportunities for future work.

II. System Model

Consider M unit-rate multi-skilled servers and N_T number of task types as depicted in Figure 1. The set of servers and task types are denoted by $\mathcal{M} = \{1, 2, \cdots, M\}$ and $\mathcal{L} = \{1, 2, \cdots, N_T\}$, respectively. Each task can be processed by any of the M servers, but with possibly different rates. The service times are assumed to be non-preemptive and discrete valued with an unknown distribution. Non-preemptive service means that the central load balancing algorithm cannot interrupt an in-service task, i.e. no other task is scheduled to a server until the server completely processes the task that is currently receiving service. The extension of the analysis for continuous service time, using approximation methods of continuous distributions with discrete ones, is an interesting future work. In this discrete time model, time is indexed by $t \in \mathbb{N}$. In the following, service time distributions and task arrivals are discussed, which are both unknown to the central scheduler.

Service Time Distribution: The service time offered by server $m \in \mathcal{M}$ to task type $i \in \mathcal{L}$ is a discrete-type random variable with cumulative distribution function (CDF) $F_{i,m}$ with mean $\frac{1}{\mu_{i,m}}$ or correspondingly with rate $\mu_{i,m} > 0$. The service time distribution does not require the memoryless property. We further assume that the support of the service time is bounded, which is a realistic assumption and reduces the unnecessary complexity of the proofs specially in Lemma 4. The extension of the analysis for service times with unbounded supports is an interesting future work. Note that the completion time for a task is the waiting time for that task until it is scheduled to a server plus the service time of the task on the server. Waiting time depends on the servers' status, the queue lengths or more specifically other tasks that are in the system or may arrive later, and the load balancing algorithm that is used, while service time has the mentioned distribution.

Task Arrival: The number of incoming tasks of type $i \in \mathcal{L}$ at the beginning of time slot t is a random variable on non-negative integer numbers that is denoted by $A_i(t)$, which are temporarily identically distributed and independent from each other. Denote the arrival rate of task type i by λ_i, i.e. $\mathbb{E}[A_i(t)] = \lambda_i$. In the stability proof of Blind GB-PANDAS we need λ_i to be strictly positive, so without loss of generality we exclude task types with zero arrival rate from $\mathcal{L}$. Furthermore, we assume that the number of each incoming task type at a time slot is bounded by constant C_A and is zero with positive probability, i.e. $P(A_i(t) < C_A) = 1$ and $P(A_i(t) = 0) > 0$ for any $i \in \mathcal{L}$. The set of arrival rates for all task types is denoted by vector $\boldsymbol{\lambda} = (\lambda_i : i \in \mathcal{L})$.

Affinity scheduling problem refers to load balancing for such a system described above. The fluid model planning algorithm [3], MaxWeight [8], and cμ-rule [9] are the baseline algorithms for affinity scheduling. All these algorithms in addition to GB-PANDAS use the rate of service times instead of the CDF functions. Hence, the system model can be summarized as an $N_T \times M$ matrix, where element (i, m) is the processing rate of task type i on server m, $\mu_{i,m}$, as follows:

$$B_\mu = \begin{bmatrix} \mu_{1,1} & \mu_{1,2} & \mu_{1,3} & \cdots & \mu_{1,M} \\ \mu_{2,1} & \mu_{2,2} & \mu_{2,3} & \cdots & \mu_{2,M} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mu_{N_T,1} & \mu_{N_T,2} & \mu_{N_T,3} & \cdots & \mu_{N_T,M} \end{bmatrix}_{N_T,M} . \quad (1)$$

If both the set of arrival rates $\boldsymbol{\lambda} = (\lambda_i : i \in \mathcal{L})$ and the service rate matrix B_μ are known, the fluid model planning algorithm [3] derives the delay optimal load balancing by solving a linear programming. However, if the arrival rates of task types are not known, the delay optimal algorithm becomes an open problem which has not been solved for more than three decades. Max-Weight [8] and cμ-rule [9] can be used for different objectives when we do not know the arrival rates, but none have delay optimality. In this work, we are assuming that we lack knowledge of not only the arrival rates $\boldsymbol{\lambda}$, but also the service rate matrix B_μ. We take an exploration and exploitation approach to make our estimation of the underlying model, which is the service rate matrix, more accurate, and to keep the system stable.

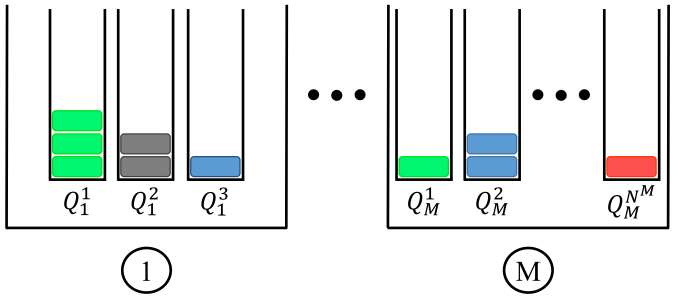

Fig. 2. The queueing structure for the GB-PANDAS algorithm.

A. Queueing Structure for GB-PANDAS

Every algorithm has its own specific queueing structure. For example, there is only a single central queue for the First-Come-First-Served (FCFS) algorithm, but there are N_T number of queues when using fluid model planning, Max-Weight, or cμ-rule. In the following, we present the queueing structure used for GB-PANDAS that captures the trade-off between the complexity of the system model and the complexity of the queueing structure very well. What we mean by the complexity of the system model is the heterogeneity of the service rate matrix, e.g. if all the elements of this matrix are the same number, the system is less complex than the case where each element of the matrix is different from other elements of the matrix.

The heterogeneity of the system from the perspective of server m is captured in the m^{th} column of the service rate matrix. Consider the m^{th} column of the matrix has N^m distinct values, where N^m can be any number from 1 to N_T. It is obvious that any of the task types with the same service (processing) rate on server m look the same from the perspective of this server. Denote the N^m distinct values of the m^{th} column of B_μ by $\{\alpha_m^1, \alpha_m^2, \cdots, \alpha_m^{N^m}\}$ and without loss of generality assume that $\alpha_m^1 > \alpha_m^2 > \cdots > \alpha_m^{N^m}$. We call all the task types with a processing rate of α_m^n on the m^{th} server, the n-local tasks to that server, and denote them by $\mathcal{L}_m^n = \{i \in \mathcal{L} : \mu_{i,m} = \alpha_m^n\}$. For ease of notation, we use both $\mu_{i,m}$ and α_m^n throughout the paper interchangeably; however, they are in fact capturing the same phenomenon, but with different interpretations. Note that the n-local tasks to server m can be called (n, m)-local tasks in order to place more emphasis on the pair n and m, so the n-local tasks to server m are not necessarily the same as the n-local tasks to server m'. We allocate N^m queues for server m, where the n^{th} queue of server m holds all task types that are routed to this server and are n-local to it. As depicted in Figure 2, different servers can have different numbers of queues since the heterogeneity of the system model can be different from the perspective of different servers. We may interchangeably use queue or sub-queue to refer to the n^{th} queue (sub-queue) of the m^{th} server. The N^m sub-queues of the m^{th} server are denoted by $Q_m^1, Q_m^2, \cdots, Q_m^{N^m}$ and the queue lengths of these sub-queues, defined as the number of tasks in these sub-queues, at time slot t are denoted by $Q_m^1(t), Q_m^2(t), \cdots, Q_m^{N^m}(t)$.

In the next subsection, the GB-PANDAS algorithm is proposed when the service rate matrix B_μ is known.

Balanced-PANDAS for a data center with three levels of data locality is proposed by [6], and here we are proposing the Generalized Balanced-PANDAS algorithm from another perspective which is of its own interest.

B. GB-PANDAS Algorithm With Known Service Rate Matrix B_μ

Before getting into the GB-PANDAS algorithm, we need to define the workload on server m.

Definition 1: The average time needed for server m to process all tasks queued in its N^m sub-queues at time slot t is defined as the workload on the server:

$$W_m(t) = \frac{Q_m^1(t)}{\alpha_m^1} + \frac{Q_m^2(t)}{\alpha_m^2} + \cdots + \frac{Q_m^{N^m}(t)}{\alpha_m^{N^m}}. \tag{2}$$

A load balancing algorithm consists of two parts, routing and scheduling. The routing policy determines the queue at which an incoming task is stored until it is assigned to a server for service. When a server becomes idle, the scheduling policy determines the next task that receives service on the idle server. The routing and scheduling policies of the GB-PANDAS algorithm are as follows:

GB-PANDAS Routing Policy: An incoming task of type i is routed to the corresponding sub-queue of the server with the minimum weighted workload, where ties are broken arbitrarily to the favor of the fastest server. The server m^* with the minimum weighted workload is defined as

$$m^* = \arg\min_{m \in \mathcal{M}} \frac{W_m(t)}{\mu_{i,m}}.$$

The corresponding sub-queue of server m^* for a task of type i is n if $\mu_{i,m} = \alpha_m^n$.

GB-PANDAS Scheduling Policy: An idle server m at time slot t is scheduled to process a task of sub-queue Q_m^1 if there is any. If $Q_m^1(t) = 0$, a task of sub-queue Q_m^2 is scheduled to the server, and so on. It is a common assumption that servers do not have the option of processing the tasks queued in front of other servers, so a server remains idle if all its sub-queues are empty. Note that the routing policy is doing a sort of weighted water-filling for workloads, so the probability that a server becomes idle goes to zero as the load increases at heavy traffic regime. Remember that the tasks in sub-queue Q_m^1 are the fastest types of tasks for server m, the tasks in sub-queue Q_m^2 are the second fastest, and so on. Using this priority scheduling, the faster tasks in the N^m sub-queues of server m are processed first. Given the minimum weighted workload routing policy, the priority scheduling is optimal as it minimizes the mean task completion time of all tasks in the N^m sub-queues of server m. In the following, Max-Weight and cμ-rule algorithms are discussed for the sake of completeness.

Remark: Prioritized scheduling has no effect in the throughput-optimality proof of the GB-PANDAS algorithm and a work conservative scheduling of a server to its sub-queues suffices for the purpose of system's stability. As a result, the GB-PANDAS policy can be implemented by considering a single queue per server at the expense of losing priority scheduling. In a single queue per server structure, instead of maintaining a server's sub-queue lengths, the workload of the server defined in (2) is maintained. At the arrival of an n-local task to server m, the server's workload is increased by $\frac{1}{\alpha_m^n}$, instead of increasing the corresponding sub-queue's length by one, and the workload is decreased at the departure of a task by its corresponding load.

C. Max-Weight and c-μ-Rule Algorithms With Known Service Rate Matrix B_μ

The queueing structure used for Max-Weight and c-μ-rule is as depicted in Figure 1, where there is a separate queue for each type of task. Denote the N_T queues by $Q_1, Q_2, \cdots, Q_{N_T}$, and their corresponding queue lengths at time slot t by $Q_1(t), Q_2(t), \cdots, Q_{N_T}(t)$. Note that the GB-PANDAS algorithm requires $M \times N_T$ number of queues in the worst case scenario, but it can use the symmetry of specific real-world structures to decrease the number of queues dramatically. As an example, for servers with rack structures, where Hadoop is used for map-reduce data placement with three replicas of data chunks on severs, Max-Weight and c-μ-rule require $\binom{M}{3} = O(M^3)$ number of queues, while GB-PANDAS requires $3M$ queues. A task is routed to a server at the time of its arrival under the GB-PANDAS algorithm, while a task waits in its queue under both Max-Weight and c-μ-rule algorithms, waiting to be scheduled for service, which is discussed below.

Max-Weight Scheduling Policy: An idle server m at time slot t is scheduled to process a task of type j from Q_j, if there is any, such that

$$j \in \arg\max_{i \in \mathcal{L}} \{\mu_{i,m} \cdot Q_i(t)\}.$$

The Max-Weight algorithm is throughput-optimal, but it is not heavy-traffic or delay optimal [8].

C-μ-Rule Scheduling Policy: Consider that queue Q_i incurs a cost of $C_i\big(Q_i(t)\big)$ at time slot t, where $C_i(.)$ is increasing and strictly convex. The c-μ-rule algorithm maximizes the rate of decrease of the instantaneous cost at all time slots by the following scheduling policy. An idle server m at time slot t is scheduled to process a task of type j from Q_j, if there is any, such that

$$j \in \arg\max_{i \in \mathcal{L}} \big\{\mu_{i,m} \cdot C_i'\big(Q_i(t)\big)\big\},$$

where $C'(.)$ is the first derivative of the cost function. The c-μ-rule algorithm minimizes both instantaneous and cumulative queueing costs, asymptotically. The mean task completion time corresponds to linear cost functions for all task types, so c-μ-rule cannot minimize the mean task completion time, and as the result, is not heavy-traffic optimal.

D. Capacity Region of Affinity Scheduling Setup

We propose a decomposition of the arrival rate vector $\boldsymbol{\lambda} = (\lambda_i : i \in \mathcal{L})$ as follows. For any task type $i \in \mathcal{L}, \lambda_i$ is decomposed into $(\lambda_{i,m}, m \in \mathcal{M})$, where $\lambda_{i,m}$ is assumed to be the arrival rate of type i tasks for server m. Hence, $\lambda_i = \sum_{m=1}^{M} \lambda_{i,m}$. By using the fluid model planning algorithm, the affinity queueing system can be stabilized under a given arrival rate vector $\boldsymbol{\lambda}$ as long as the necessary condition of total 1-local, 2-local, …, and N^m-local load on server m being strictly less than one for any server m is satisfied:

$$\sum_{i \in \mathcal{L}} \frac{\lambda_{i,m}}{\mu_{i,m}} < 1, \quad \forall m \in \mathcal{M}. \tag{3}$$

Hence, the capacity region of the affinity problem is the set of all arrival rate vectors $\boldsymbol{\lambda}$ that has a decomposition $(\lambda_{i,m}, i \in \mathcal{L}, m \in \mathcal{M})$ satisfying (3):

$$\Lambda = \Big\{ \boldsymbol{\lambda} = (\lambda_i : i \in \mathcal{L}) \;\Big|\; \exists \lambda_{i,m} \geq 0, \forall i \in \mathcal{L}, \quad \forall m \in \mathcal{M}, s.t. \\ \lambda_i = \sum_{m=1}^{M} \lambda_{i,m}, \ \forall i \in \mathcal{L}, \quad \sum_{i \in \mathcal{L}} \frac{\lambda_{i,m}}{\mu_{i,m}} < 1, \quad \forall m \in \mathcal{M} \Big\}. \quad (4)$$

The linear programming optimization associated with Equation (4) can be solved to find the capacity region Λ of the system. The GB-PANDAS algorithm stabilizes the system for any arrival rate vector inside the capacity region by knowing the service rate matrix. It is proved in Section IV that the Blind GB-PANDAS algorithm is throughput-optimal without the knowledge of the service rate matrix, B_μ.

III. The Blind GB-PANDAS Algorithm

The GB-PANDAS and Max-Weight algorithms need to know the precise value of the service rate matrix, but this requirement is not realistic for real applications. Furthermore, the service rate matrix can change over time, which confuses the load balancing algorithm if it uses a fixed given service rate matrix. In the Blind version of GB-PANDAS, the service rate matrix is initiated randomly and is updated as the system is running. More specifically, an exploration-exploitation framework is combined with GB-PANDAS. In the exploration phase, the routing and scheduling are performed so as to allow room for making the estimations of the system parameters more precise, and in the exploitation phase the routing and scheduling are done based on the available estimation of the service rate matrix so as to stabilize the system. Here we assume that N^m is known as well as the locality level of a task on servers that can be inferred from prior knowledge on the structure of the system. This is not a necessary assumption for throughput-optimality proof, but it makes the intuition behind Blind GB-PANDAS more clear. As mentioned before, a single queue per server can be used when using Blind GB-PANDAS, in which case, there is no need to know N^m as well as the ordering of service rates offered by servers for different task types.

We first propose the updating method used for the service rate matrix before getting into the routing and scheduling policies of the Blind GB-PANDAS algorithm. The estimated service rate matrix at time slot t is denoted as

$$\widetilde{B}_\mu(t) = \begin{bmatrix} \widetilde{\mu}_{1,1}(t) & \widetilde{\mu}_{1,2}(t) & \widetilde{\mu}_{1,3}(t) & \cdots & \widetilde{\mu}_{1,M}(t) \\ \widetilde{\mu}_{2,1}(t) & \widetilde{\mu}_{2,2}(t) & \widetilde{\mu}_{2,3}(t) & \cdots & \widetilde{\mu}_{2,M}(t) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \widetilde{\mu}_{N_T,1}(t) & \widetilde{\mu}_{N_T,2}(t) & \widetilde{\mu}_{N_T,3}(t) & \cdots & \widetilde{\mu}_{N_T,M}(t) \end{bmatrix}. \quad (5)$$

Note that $\widetilde{\alpha}_m^1(t), \widetilde{\alpha}_m^2(t), \cdots, \widetilde{\alpha}_m^{N^m}(t), \ \forall m \in \mathcal{M}$ which are the estimates of $\alpha_m^1(t), \alpha_m^2(t), \cdots, \alpha_m^{N^m}(t), \ \forall m \in \mathcal{M}$ at time slot t are nothing but the distinct values of the elements of the service rate matrix. More specifically, those are the $\widetilde{\alpha}_m^n, \ \forall m \in \mathcal{M}, \ \forall n \in \{1, 2, \cdots, N^m\}$ that are getting updated and then mapped into their corresponding elements in the service rate matrix to form $\widetilde{B}_\mu$ in (5) as mentioned in Section II-A. Consider a random initialization of $\widetilde{\alpha}_m^n(0) > 0$, $\forall m \in \mathcal{M}, \ \forall n \in \{1, 2, \cdots, N^m\}$ at time slot $t = 0$. If server m has processed $\widetilde{n} - 1$ tasks that are n-local to this server by time t_1, the estimate of α_m^n at this time slot is $\widetilde{\alpha}_m^n(t_1)$, and a new observation of service time for n-local task to server m is made at time slot $t_2 > t_1$ as $T_m^n(t_2)$, we have $\widetilde{\alpha}_m^n(t) = \widetilde{\alpha}_m^n(t_1)$ for $t_1 \leq t < t_2$ and the update of this parameter at time slot t_2 is

$$\widetilde{\alpha}_m^n(t_2) = \frac{\widetilde{n} - 1}{\widetilde{n}} \cdot \widetilde{\alpha}_m^n(t_1) + \frac{1}{\widetilde{n} \cdot T_m^n(t_2)}. \quad (6)$$

Note that $\widetilde{\alpha}_m^n$ is the service rate, not the service time mean, that is why $\frac{1}{T_m^n(t_2)}$ is used above in the update of the service rate. In the following, the routing and scheduling policies of Blind GB-PANDAS are presented, where the exploration rate is chosen in such a way that infinitely many n-local tasks are scheduled for service on server m for any $m \in \mathcal{M}$ and any $n \in \{1, 2, \cdots, N^m\}$ so that by using the strong law of large numbers, the parameter estimations in (6) converge to their real values almost surely.

Blind GB-PANDAS Routing Policy: The estimated workload on server m at time slot t is defined based on parameter estimations in (6) as

$$\widetilde{W}_m(t) = \frac{Q_m^1(t)}{\widetilde{\alpha}_m^1(t)} + \frac{Q_m^2(t)}{\widetilde{\alpha}_m^2(t)} + \cdots + \frac{Q_m^{N^m}(t)}{\widetilde{\alpha}_m^{N^m}(t)}. \quad (7)$$

The routing of an incoming task is based on the following exploitation policy with probability $p_e = max(1 - p(t), 0)$, and is based on the exploration policy otherwise, where $p(t) \to 0$ as $t \to \infty$ and $\sum_{t=0}^{\infty} p(t) = \infty$, e.g. the exploitation probability can be chosen as $p_e = 1 - \frac{1}{t^c}$ for $0 < c \leq 1$.

- *Exploitation Phase:* An incoming task of type i is routed to the corresponding sub-queue of the server with the minimum estimated weighted workload, where ties are broken arbitrarily. The server $\widetilde{m}^*$ with the minimum weighted workload for task of type i is defined as
$$\widetilde{m}^* = \underset{m \in \mathcal{M}}{\arg\min} \frac{\widetilde{W}_m(t)}{\widetilde{\mu}_{i,m}(t)}.$$
The corresponding sub-queue of server $\widetilde{m}^*$ for a task of type i is n if $\widetilde{\mu}_{i,\widetilde{m}^*} = \widetilde{\alpha}_{\widetilde{m}^*}^n$.
- *Exploration Phase:* An incoming task of type i is routed to the corresponding sub-queue of a server chosen uniformly at random among $\{1, 2, \cdots, M\}$.

Blind GB-PANDAS Scheduling Policy: The scheduling of an idle server is based on the following exploitation policy with probability p_e, and is based on the exploration policy otherwise.

- *Exploitation Phase:* Priority scheduling is performed for an idle server as discussed in Section II-B. We emphasize that given the routing policy, priority scheduling is the optimal scheduling policy in terms of minimizing the average completion time of tasks.
- *Exploration Phase:* An idle server is scheduled to one of its non-empty sub-queues uniformly at random, and stays idle if all its sub-queues are empty.

Since the arrival rate of any task type is strictly positive, infinitely many of each task type arrives to system, and given the fact that the probability of exploration in both routing and scheduling policies decays such that $\sum_{t=0}^{\infty} p(t) = \infty$, using the second Borel-Cantelli lemma (zero-one law), it is obvious that n-local tasks to server m are scheduled to this server for infinitely many times for any locality level and any server, so $\widetilde{B}_\mu(t) \to B_\mu$ as $t \to \infty$ using the updates in (6).

A. Queue Dynamics Under the Blind GB-PANDAS Algorithm

Denote the queue length vector at time slot t by $\boldsymbol{Q}(t) = \big(Q_1^1(t), Q_1^2(t), \cdots, Q_1^{N^1}(t), \cdots, Q_M^{N^M}(t)\big)$. Let the number of incoming tasks of type i that are routed to their corresponding sub-queue of server m at the beginning of time slot t be denoted as $A_{i,m}(t)$. Then, by denoting the number of incoming n-local tasks to server m that are routed to Q_m^n at the beginning of time slot t by $A_m^n(t)$, we have:

$$A_m^n(t) = \sum_{i \in \mathcal{L}_m^n} A_{i,m}(t),\ \forall m \in \mathcal{M},\ 1 \le n \le N^m. \tag{8}$$

Denote the set of working status of servers by vector $\boldsymbol{f}(t) = \big(f_1(t),\ f_2(t),\ \cdots,\ f_M(t)\big)$, where

$$f_m(t) = \begin{cases} -1, & \text{if server } m \text{ is idle,} \\ 1, & \text{if server } m \text{ processes a 1-local task from } Q_m^1, \\ 2, & \text{if server } m \text{ processes a 2-local task from } Q_m^2, \\ \vdots & \\ N^m, & \text{if server } m \text{ processes an } N^m\text{-local task} \\ & \quad \text{from } Q_m^{N^m}. \end{cases}$$

If server m finishes processing a task at the end of time slot $t-1$, i.e. $f_m(t^-) = -1$, a scheduling decision is taken for time t based on $\boldsymbol{Q}(t)$ and $\boldsymbol{f}(t)$. Denote the scheduling decision for server m at time slot t by $\eta_m(t)$ that is defined as follows. For all busy servers, $\eta_m(t) = f_m(t)$, and when $f_m(t^-) = -1$, i.e. server m is idle, $\eta_m(t)$ is determined by the scheduler according to the Blind GB-PANDAS algorithm. Let $\boldsymbol{\eta}(t) = \big(\eta_1(t),\ \eta_2(t), \cdots, \eta_M(t)\big)$.

Let $S_m^n(t)$ denote the n-local service provided by server m, where such a service has the rate of α_m^n if $\eta_m(t) = n$ for $1 \le n \le N^m$, and the rate is zero otherwise. Then, the queue dynamics for any $m \in \mathcal{M}$ is as follows:

$$\begin{aligned} Q_m^n(t+1) &= Q_m^n(t) + A_m^n(t) - S_m^n(t), \quad \text{for } 1 \le n \le N^m - 1, \\ Q_m^{N^m}(t+1) &= Q_m^{N^m}(t) + A_m^{N^m}(t) - S_m^{N^m}(t) + U_m(t), \end{aligned} \tag{9}$$

where $U_m(t) = \max\big\{0, S_m^{N^m}(t) - A_m^{N^m}(t) - Q_m^{N^m}(t)\big\}$ is the unused service offered by server m at time slot t.

Note that $\big\{\boldsymbol{Q}(t),\ t \ge 0\big\}$ does not necessarily form a Markov chain, i.e. $\boldsymbol{Q}(t+1) | \boldsymbol{Q}(t) \not\perp \boldsymbol{Q}(t-1)$, since nothing can be said about locality of an in-service task at a server by just knowing the queue lengths. Even $\big\{\ (\boldsymbol{Q}(t), \boldsymbol{\eta}(t))\ ,\ t \ge 0\big\}$ is not a Markov chain since the service time distributions do not necessarily have the memory-less property. In order to use Foster-Lyapunov theorem for proving the positive recurrence of a Markov chain, we need to consider another measurement of the status of servers as follows.

- Let $\Psi_m(t)$ denote the number of time slots at the beginning of time slot t that server m has been allocated on the current in-service task on server m. This parameter is set to zero when server m finishes processing a task. Let $\boldsymbol{\Psi}(t) = \big(\Psi_1(t), \Psi_2(t), \cdots, \Psi_M(t)\big)$.

Lemma 1: $\big\{\boldsymbol{Z}(t) = \big(\boldsymbol{Q}(t), \boldsymbol{\eta}(t), \boldsymbol{\Psi}(t)\big), t \ge 0\big\}$ forms an irreducible and aperiodic Markov chain. The state space of this Markov chain is $\mathcal{S} = \big(\prod_{m \in \mathcal{M}} \mathbb{N}^{N^m}\big) \times \big(\prod_{m \in \mathcal{M}} \{1, 2, \cdots, N^m\}\big) \times \mathbb{N}^M$.

IV. Throughput Optimality of the Blind GB-PANDAS Algorithm

Section IV-A provides preliminaries on the workload dynamic of servers, the ideal workload on servers, some lemmas, and an extended version of the Foster-Lyapunov. The throughput-optimality theorem of the Blind GB-PANDAS algorithm and its proof are presented in Section IV-B, where the proof is followed by using Lemmas 2, 3, 4, 5, and 6.

A. Preliminary Materials and Lemmas

The workload on server m evolves as follows:

$$\begin{aligned} &W_m(t+1) \\ &= \frac{Q_m^1(t+1)}{\alpha_m^1} + \frac{Q_m^2(t+1)}{\alpha_m^2} + \cdots + \frac{Q_m^{N^m}(t+1)}{\alpha_m^{N^m}} \\ &\overset{(a)}{=} \frac{Q_m^1(t) + A_m^1(t) - S_m^1(t)}{\alpha_m^1} + \frac{Q_m^2(t) + A_m^2(t) - S_m^2(t)}{\alpha_m^2} \\ &\quad + \cdots + \frac{Q_m^{N^m}(t) + A_m^{N^m}(t) - S_m^{N^m}(t) + U_m(t)}{\alpha_m^{N^m}} \\ &= W_m(t) + \left(\frac{A_m^1(t)}{\alpha_m^1} + \frac{A_m^2(t)}{\alpha_m^2} + \cdots + \frac{A_m^{N^m}(t)}{\alpha_m^{N^m}}\right) \\ &\quad - \left(\frac{S_m^1(t)}{\alpha_m^1} + \frac{S_m^2(t)}{\alpha_m^2} + \cdots + \frac{S_m^{N^m}(t)}{\alpha_m^{N^m}}\right) + \frac{U_m(t)}{\alpha_m^{N^m}} \\ &\overset{(b)}{=} W_m(t) + A_m(t) - S_m(t) + \widetilde{U}_m(t), \end{aligned}$$

where (a) is true by using the queue dynamics in (9) and (b) follows from defining the pseudo task arrival, service, and unused services of server m as

$$\begin{aligned} A_m(t) &= \frac{A_m^1(t)}{\alpha_m^1} + \frac{A_m^2(t)}{\alpha_m^2} + \cdots + \frac{A_m^{N^m}(t)}{\alpha_m^{N^m}},\ \forall m \in \mathcal{M}, \\ S_m(t) &= \frac{S_m^1(t)}{\alpha_m^1} + \frac{S_m^2(t)}{\alpha_m^2} + \cdots + \frac{S_m^{N^m}(t)}{\alpha_m^{N^m}},\ \forall m \in \mathcal{M}, \\ \widetilde{U}_m(t) &= \frac{U_m(t)}{\alpha_m^{N^m}}, \quad \forall m \in \mathcal{M}. \end{aligned} \tag{10}$$

By defining the pseudo task arrival, service, and unused service processes as $\boldsymbol{A}(t) = \big(A_1(t), A_2(t), \cdots, A_M(t)\big)$, $\boldsymbol{S}(t) = \big(S_1(t), S_2(t), \cdots, S_M(t)\big)$, and $\widetilde{\boldsymbol{U}}(t) = \big(\widetilde{U}_1(t), \widetilde{U}_2(t), \cdots, \widetilde{U}_M(t)\big)$, respectively, the vector of servers' workloads defined by $\boldsymbol{W} = (W_1, W_2, \cdots, W_M)$ evolves as

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) + \boldsymbol{A}(t) - \boldsymbol{S}(t) + \widetilde{\boldsymbol{U}}(t). \tag{11}$$

Lemma 2: For any arrival rate vector inside the capacity region, $\boldsymbol{\lambda} \in \Lambda$, there exists a load decomposition $\{\lambda_{i,m}\}$ and $\delta > 0$ such that

$$\sum_{i \in \mathcal{L}} \frac{\lambda_{i,m}}{\mu_{i,m}} < \frac{1}{1+\delta}, \quad \forall m \in \mathcal{M}. \tag{12}$$

The fluid model planning algorithm solves a linear programming to find the load decomposition $\{\lambda_{i,m}\}$ that is used in its load balancing on the M servers. In other words, this load decomposition is a possibility of task assignment on servers to stabilize the system.
Lemma 2 is used in the proof of Lemma 5.

Definition 2: The ideal workload on server m corresponding to the load decomposition $\{\lambda_{i,m}\}$ of Lemma 2 is defined as

$$w_m = \sum_{i\in\mathcal{L}} \frac{\lambda_{i,m}}{\mu_{i,m}}, \quad \forall m \in \mathcal{M}. \tag{13}$$

Let $\boldsymbol{w} = (w_1, w_2, \cdots, w_M)$. The vector of servers' ideal workload is used as an intermediary term in Lemmas 4 and 5 which are later used for throughput optimality proof of the Blind GB-PANDAS algorithm.

Lemma 3:

$$\langle \boldsymbol{W}(t), \widetilde{\boldsymbol{U}}(t)\rangle = 0, \quad \forall t.$$

The following lemma states that the sum over a time period of the inner product of the workload and the pseudo arrival rate is dominated on an expectation sense by the inner product of the workload and the ideal workload plus constants depending on the initial state of the system.

Lemma 4: Under the exploration-exploitation routing policy of the Blind GB-PANDAS algorithm, for any arrival rate vector inside the capacity region, $\boldsymbol{\lambda} \in \Lambda$, and the corresponding ideal workload vector $\boldsymbol{w}$ defined in (13), and for any arbitrary small $\theta_0 > 0$, there exists $T_0 > t_0$ such that for any $t_0 \geq 0$ and $T > T_0$:

$$\mathbb{E}\Big[\sum_{t=T_0}^{t_0+T-1} \Big(\langle \boldsymbol{W}(t), \boldsymbol{A}(t)\rangle - \langle \boldsymbol{W}(t), \boldsymbol{w}\rangle\Big)\Big|\boldsymbol{Z}(t_0)\Big] \leq \theta_0\, T\|\boldsymbol{Q}(t_0)\|_1 + c_0,$$

where the constants $\theta_0, c_0 > 0$ are independent of $\boldsymbol{Z}(t_0)$.

We emphasize that θ_0 in Lemma 4 can be made arbitrarily small, as can be seen in the proof, which is used in the throughput optimality proof of Blind GB-PANDAS, Theorem 1. Throughout this paper, $\|.\|$ and $\|.\|_1$ are the L^2-norm and L^1-norm, respectively.

The following lemma is the counterpart of Lemma 4 for the pseudo service process.

Lemma 5: Under the exploration-exploitation scheduling policy of the Blind GB-PANDAS algorithm, for any arrival rate vector inside the capacity region, $\boldsymbol{\lambda} \in \Lambda$, and the corresponding ideal workload vector $\boldsymbol{w}$ in (13), there exists $T_1 > 0$ such that for any $T > T_1$, we have:

$$\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1} \Big(\langle \boldsymbol{W}(t), \boldsymbol{w}\rangle - \langle \boldsymbol{W}(t), \boldsymbol{S}(t)\rangle\Big)\Big|\boldsymbol{Z}(t_0)\right] \leq -\theta_1\, T\|\boldsymbol{Q}(t_0)\|_1 + c_1, \quad \forall t_0 \geq 0, \tag{14}$$

where the constants $\theta_1, c_1 > 0$ are independent of $\boldsymbol{Z}(t_0)$.

Lemma 6: Under the exploration-exploitation load balancing of the Blind GB-PANDAS algorithm, for any arrival rate vector inside the capacity region, $\boldsymbol{\lambda} \in \Lambda$, and for any $\theta_2 > 0$, there exists $T_2 > 0$ such that for any $T > T_2$ and for any $t_0 \geq 0$, we have:

$$\mathbb{E}\Big[\|\boldsymbol{\Psi}(t_0+T)\|_1 - \|\boldsymbol{\Psi}(t_0)\|_1\Big|\boldsymbol{Z}(t_0)\Big] \leq -\theta_2\|\boldsymbol{\Psi}(t_0)\|_1 + MT.$$

Theorem 3.3.8 in [13], an extended version of the Foster-Lyapunov theorem: Consider an irreducible Markov chain $\{\boldsymbol{Z}(t)\}$, where $t \in \mathbb{N}$, with a state space $\mathcal{S}$. If there exists a function $V : \mathcal{S} \to \mathcal{R}^+$, a positive integer $T \geq 1$, and a finite set $\mathcal{P} \subseteq \mathcal{S}$ satisfying the following condition:

$$\mathbb{E}\big[V(Z(t_0+T)) - V(Z(t_0))\big|\boldsymbol{Z}(t_0) = z\big] \leq -\theta\mathbb{I}_{\{z\in\mathcal{P}^c\}} + C\mathbb{I}_{\{z\in\mathcal{P}\}}, \tag{15}$$

for some $\theta > 0$ and $C < \infty$, then the irreducible Markov chain $\{\boldsymbol{Z}(t)\}$ is positive recurrent.

B. Throughput Optimality Theorem and Proof

Theorem 1: The Blind GB-PANDAS algorithm is throughput-optimal for a system with affinity setup discussed in Section II, with general service time distributions with finite means and supports, without prior knowledge on the service rate matrix B_μ and the arrival rate vector $\boldsymbol{\lambda}$.

Proof: We use the Foster-Lyapunov theorem for proving that the irreducible and aperiodic Markov chain $\big\{\boldsymbol{Z}(t) = \big(\boldsymbol{Q}(t), \boldsymbol{\eta}(t), \boldsymbol{\Psi}(t)\big),\ t \geq 0\big\}$ (Lemma 1) is positive recurrent under the Blind GB-PANDAS algorithm, as far as the arrival rate vector is inside the capacity region, $\boldsymbol{\lambda} \in \Lambda$. This means that as time goes to infinity, the distribution of $\boldsymbol{Z}(t)$ converges to its stationary distribution, which implies that the system is stable and Blind GB-PANDAS is throughput-optimal. To this end, we choose the following Lyapunov function $V : \mathcal{S} \to \mathcal{R}^+$ and use Lemmas 2, 3, 4, 5, and 6 to derive its drift afterward:

$$V(\boldsymbol{Z}(t)) = \|\boldsymbol{W}(t)\|^2 + \|\boldsymbol{\Psi}(t)\|_1. \tag{16}$$

By choosing θ_0 in Lemma 4 less than θ_1 in Lemma 5, $\theta_0 < \theta_1$, we get T_0 from Lemma 4, which is used in the drift of the Lyapunov function in Lemma 7.

Lemma 7: For any $t_0 \leq T_0 < T$, specifically T_0 from Lemma 4 that is dictated by choosing $\theta_0 < \theta_1$, we have the following for the drift of the Lyapunov function in (16), where T_0 is used in the first summation after the inequality:

$$\begin{aligned}
&\mathbb{E}\Big[V(\boldsymbol{Z}(t_0+T)) - V(\boldsymbol{Z}(t_0))\Big|\boldsymbol{Z}(t_0)\Big] \\
&\leq 2\mathbb{E}\left[\sum_{t=T_0}^{t_0+T-1} \Big(\langle \boldsymbol{W}(t), \boldsymbol{A}(t)\rangle - \langle \boldsymbol{W}(t), \boldsymbol{w}\rangle\Big)\Big|\boldsymbol{Z}(t_0)\right] \\
&\quad + 2\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1} \Big(\langle \boldsymbol{W}(t), \boldsymbol{w}\rangle - \langle \boldsymbol{W}(t), \boldsymbol{S}(t)\rangle\Big)\Big|\boldsymbol{Z}(t_0)\right] \\
&\quad + \mathbb{E}\Big[\|\boldsymbol{\Psi}(t_0+T)\|_1 - \|\boldsymbol{\Psi}(t)\|_1\Big|\boldsymbol{Z}(t_0)\Big] \\
&\quad + c_2\|\boldsymbol{Q}(t_0)\|_1 + c_3.
\end{aligned} \tag{17}$$

By choosing $T > \max\{T_0, T_1, T_2, \frac{\theta_2+c_2}{2(\theta_1-\theta_0)}\}$, where $\theta_2 > 0$ is the one in Lemma 6, and substituting the terms on the right-hand side of the Lyapunov function drift (17) in Lemma 7 from the corresponding inequalities in Lemmas 4, 5, and 6, we have:

$$\mathbb{E}\Big[V(\boldsymbol{Z}(t_0+T)) - V(\boldsymbol{Z}(t_0))\Big|\boldsymbol{Z}(t_0)\Big] \leq -\theta_2\Big(\|\boldsymbol{Q}(t_0)\|_1 + \|\boldsymbol{\Psi}(t_0)\|_1\Big) + c,\ \forall t_0,$$

where $c = 2c_0 + 2c_1 + c_3 + MT$.

Let $\mathcal{P} = \big\{\boldsymbol{Z} = \big(\boldsymbol{Q}, \boldsymbol{\eta}, \boldsymbol{\Psi}\big) \in \mathcal{S} : \|\boldsymbol{Q}\|_1 + \|\boldsymbol{\Psi}\|_1 \leq \frac{\bar{c}+c}{\theta_2}\big\}$ for any positive constant $\bar{c} > 0$, where $\mathcal{P}$ is a finite set of the state space $\mathcal{S}$. By this choice of $\mathcal{P}$ for the Lyapunov function $V(.)$ defined in (16), all the conditions of the Foster-Lyapunov theorem are satisfied, which completes the throughput optimality proof for the Blind GB-PANDAS algorithm.

Note that the priority scheduling in the exploitation phase of the Blind GB-PANDAS algorithm is not used for the throughput optimality proof since the expected workload of a server is decreased in the same rate no matter what locality level is receiving service from the server. As long as an idle

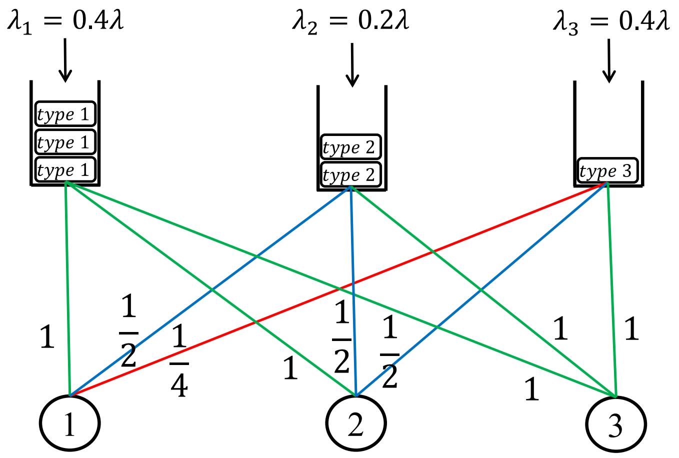

Fig. 3. The affinity structure used for simulation with three types of tasks and three multi-skilled servers.

server gives service to one of the tasks in its sub-queues continuously, the system is stable. Given the routing policy, the priority scheduling is used in the exploitation phase to minimize the mean task completion time.

V. Simulation Results

In this section, the simulated performance of the Blind GB-PANDAS algorithm is compared with FCFS, Max-Weight, and c-μ-rule algorithms. FCFS does not use system parameters for load balancing, but Max-Weight and c-μ-rule use the same exploration-exploitation approach as Blind GB-PANDAS. Convex cost functions $C_i(Q_i) = Q_i^{1.01}$ for $i \in \{1,2,3\}$ are used for the c-μ-rule algorithm. Since the objective is to minimize the mean task completion time, the convexity of the cost functions are chosen in a way to be close to a line for small values of Q_i. Three types of tasks and a computing cluster of three servers are considered with processing rates depicted in Figure 3, which are not known from the perspective of the load balancing algorithms. The task arrivals are Poisson processes with the unknown rates determined in Figure 3 and the processing times are log-normal that are heavy-tailed and do not have the memory-less property. Note that this affinity structure does not have the rack structure mentioned in [6] since from the processing rates of task type 2 on the three servers, servers 1 and 2 are in the same rack as server 3, but from the processing rates of task type 3 on the three servers, the second server is in the same rack as the third server, but not the first server. Hence, this affinity setup is more complicated than the one with a rack structure.

Inspired by the fluid model planning algorithm, the following linear programming optimization should be solved to find the capacity region of the simulated system.

$$\begin{aligned} &\underset{\lambda_{i,m}}{\text{maximize}}\ \lambda = \sum_{i=1}^{3}\sum_{m=1}^{3}\lambda_{i,m}\\ &\text{subject to: } \lambda_{1,1}+2\lambda_{2,1}+4\lambda_{3,1}<1,\ \lambda_{1,1}+\lambda_{1,2}+\lambda_{1,3}=0.4\lambda,\\ &\qquad \lambda_{1,2}+2\lambda_{2,2}+2\lambda_{3,2}<1,\ \lambda_{2,1}+\lambda_{2,2}+\lambda_{2,3}=0.2\lambda,\\ &\qquad \lambda_{1,3}+\lambda_{2,3}+\lambda_{3,3}<1,\ \lambda_{3,1}+\lambda_{3,2}+\lambda_{3,3}=0.4\lambda,\\ &\qquad \lambda_{i,m}\geq 0,\quad \forall i,m\in\{1,2,3\}. \end{aligned}$$

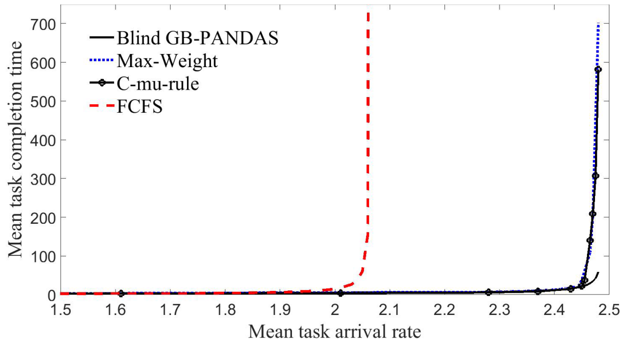

Fig. 4. Capacity region comparison of the Blind GB-PANDAS, Max-Weight, c-μ-rule, and FCFS algorithms.

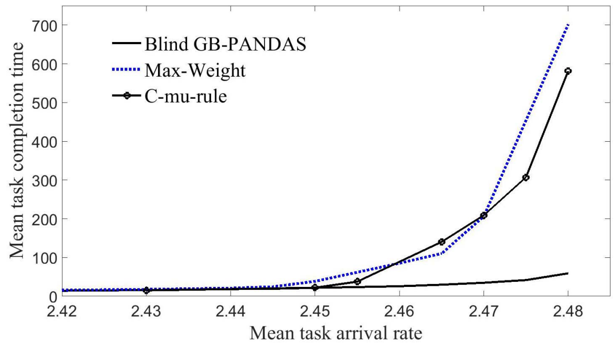

Fig. 5. Heavy-traffic performance comparison.

The capacity region in terms of λ is found to be $\lambda \in [0, 2.5)$. Figure 4 compares the throughput performance of the four algorithms, where the mean task completion time versus the total task arrival rate, $\lambda = \sum_{i=1}^{3}\lambda_i$, is plotted. The Blind GB-PANDAS, Max-Weight, and c-μ-rule algorithms are throughput-optimal by stabilizing the system for $\lambda < 2.5$. Taking a closer look at the performance of these algorithms at high loads, Blind GB-PANDAS has a much lower mean task completion time compared to Max-Weight and c-μ-rule algorithms as depicted in Figure 5.

VI. Related Work

In addition to the fluid model planning, Max-Weight, and c-μ-rule algorithms for the affinity scheduling problem that are discussed in Section I and Section II-C, there is a huge body of work on heuristic algorithms that are used for scheduling for data centers with multiple levels of data locality, e.g. [14]–[16], and look at the references in [11]. Although some of these heuristic algorithms are being used in real applications, simple facts about their optimality are not investigated. Recent works including the priority algorithm [17], Join-the-Shortest-Queue-Max-Weight (JSQ-MW) [10] and Weighted-Workload algorithm [6] study the capacity region and throughput optimality for a system with two and three levels of data locality. Some robust policies are studied in [18]–[22].

A related direction of work on scheduling for data centers with multi-level data locality, which is a direct application of affinity scheduling, is to efficiently do data replication on servers in MapReduce framework to increase availability. Increasing the availability is translated to increasing service rates in the context of this article which enlarges the capacity region and reduces the mean task completion time. For more information on data replication algorithms refer to Google File System [23], Hadoop Distributed File System [14], Scarlett [24], and Dare [25]. Data replication algorithms are complementary and orthogonal to throughput and delay optimality that is studied in this article.

Fairness is an issue in most scheduling problems which conflicts with delay optimality. A delay optimal load balancing algorithm can cooperate with fair scheduling strategies though by compromising on delay optimality to partly achieve fairness. Refer to Delay Scheduling [16], Quibcy [15], and the references therein for more details.

VII. Conclusion and Future Work

The Blind GB-PANDAS algorithm is proposed for the affinity load balancing problem where no knowledge of the task arrival rates and the service rate matrix is available. An exploration-exploitation approach is proposed for load balancing which consists of exploration and exploitation phases. The system is proven to be stable under Blind GB-PANDAS and is shown empirically through simulations to have a better delay performance than Max-Weight, c-μ-rule, and FCFS algorithms. Investigating the subspace of the capacity region in which GB-PANDAS is delay optimal is a promising direction for future work. Note that both GB-PANDAS and Max-Weight algorithms have high routing and scheduling computation complexity which can be alleviated using power-of-d-choices [26] or join-idle-queue [27] algorithms which are interesting directions to study as well. Another interesting future work is to consider a case where there are precedence relations between several tasks of a job, i.e. a departing task may join another queue.

Appendix

A. Proof of Lemma 1

Consider $\boldsymbol{Z}(0) = \{0_{(\sum_{m\in\mathcal{M}} N^m)\times 1}, \prod_{m\in\mathcal{M}} N^m,\ 0_{M\times 1}\}$ as the initial state of the Markov chain $\boldsymbol{Z}(t)$.
Irreducible: Since $F_{i,m}$ is increasing for any task-server pair, we can find an integer $\tau > 0$ such that $F_{i,m}(\tau) > 0$ for any $1 \le i \le N^m$ and $m \in \mathcal{M}$. Furthermore, probability of zero task arrival is positive in each time slot. Hence, for any state $\boldsymbol{Z} = (\boldsymbol{Q}, \boldsymbol{\eta}, \boldsymbol{\Psi})$, there is a positive probability that each task receives service in τ time slots and no new task arrives at the system in $\tau \sum_{m\in\mathcal{M}} \sum_{n=1}^{N^m} Q_m^n$ time slots. Accordingly, the initial state of the Markov chain is reachable from any states of the system. Conversely, using the same approach, it is easy to see that any states of the system is reachable from the initial state, $\boldsymbol{Z}(0)$. Consequently, the Markov chain $\boldsymbol{Z}(t)$ is irreducible.

Aperiodic: Since Markov chain $\boldsymbol{Z}(t)$ is irreducible, in order to show that it is also aperiodic, it suffices to show that there is a positive probability for transition from a state to itself. Due to the fact that there is a positive probability that zero task arrives to the system, the Markov chain stays at the initial state with a positive probability. Hence, the Markov chain $\boldsymbol{Z}(t)$ is aperiodic.

B. Proof of Lemma 2

The capacity region Λ is an open set, so for any $\boldsymbol{\lambda} \in \Lambda$, there exists $\delta > 0$ such that $(1+\delta)\boldsymbol{\lambda} = \boldsymbol{\lambda}' \in \Lambda$. On that account, (4) follows by $\sum_{i\in\mathcal{L}} \frac{\lambda'_{i,m}}{\mu_{i,m}} = \sum_{i\in\mathcal{L}} \frac{(1+\delta)\lambda_{i,m}}{\mu_{i,m}} < 1, \forall m \in \mathcal{M}$, which completes the proof:

$$\sum_{i\in\mathcal{L}} \frac{\lambda_{i,m}}{\mu_{i,m}} < \frac{1}{1+\delta}, \quad \forall m \in \mathcal{M}.$$

C. Proof of Lemma 3

$$\langle \boldsymbol{W}(t), \widetilde{\boldsymbol{U}}(t)\rangle = \sum_{m\in\mathcal{M}} \left(\frac{Q_m^1(t)}{\alpha_m^1} + \frac{Q_m^2(t)}{\alpha_m^2} + \cdots + \frac{Q_m^N(t)}{\alpha_m^{N^m}} \right) \frac{U_m(t)}{\alpha_m^{N^m}}.$$

If the unused service for server m is zero, $U_m(t) = 0$, the corresponding term for server m is zero in the above summation. Alternatively, the unused service of server m is positive if and only if all N^m sub-queues of the server are empty, which again makes the corresponding term for server m in the above summation equal to zero.

D. Proof of Lemma 4

By the choice of exploration rate for Blind GB-PANDAS, which is independent of the system state, and the fact that exploration exists in both routing and scheduling, any task that is n-local to server m is scheduled on this server for infinitely many times in the interval $[t_0, \infty)$ only due to exploration, regardless of the initial system state. Processing time of an n-local task on server m has a finite mean. Hence, due to strong law of large numbers, using the update rule (6) for the elements of the service rate matrix, we have:

$$\forall\ 0 < \epsilon < \frac{1}{2} \times \min\big\{ \min_{n\neq n',m} \big|\alpha_m^n - \alpha_m^{n'}\big|, \min_{m,n} \alpha_m^n, 0.5 \big\}$$
and $\forall \delta' > 0, \exists T_0' > t_0,$ such that for any $\boldsymbol{Z}(t_0)$
$$P\left(\big|\widetilde{\alpha}_m^n(t) - \alpha_m^n\big| < \epsilon,\ 1-\epsilon < \frac{\alpha_m^n}{\widetilde{\alpha}_m^n(t)} < 1+\epsilon \,\middle|\, \boldsymbol{Z}(t_0) \right) > 1 - \delta',$$
$$\forall t > T_0',\ \forall m \in \mathcal{M},\ \forall n \in \{1, 2, \cdots, N^m\}. \tag{18}$$

By the above choice of ϵ, for $t > T_0'$, the different locality levels are distinct from each other with at least $1 - \delta'$ probability. Let E be the event that $\big|\widetilde{\alpha}_m^n(t) - \alpha_m^n\big| < \epsilon$ and $1 - \epsilon < \frac{\alpha_m^n}{\widetilde{\alpha}_m^n(t)} < 1 + \epsilon$ for $t \ge T_0'$.

For an incoming task of type $i \in \mathcal{L}$ at time slot t, define the *exact* (but not known) and *estimated* minimum weighted workloads as

$$\overline{W}_i^*(t) = \min_{m\in\mathcal{M}} \frac{W_m(t)}{\mu_{i,m}}, \quad \widetilde{\overline{W}}_i^*(t) = \min_{m\in\mathcal{M}} \frac{\widetilde{W}_m(t)}{\widetilde{\mu}_{i,m}(t)}, \tag{19}$$

where $W_m(t)$ and $\widetilde{W}_m(t)$ are defined in (2) and (7), respectively. $W_m(t)$ and $\widetilde{W}_m(t)$ are related to each other as follows:

$$\begin{aligned}\widetilde{W}_m(t) &= \frac{Q_m^1(t)}{\widetilde{\alpha}_m^1(t)} + \frac{Q_m^2(t)}{\widetilde{\alpha}_m^2(t)} + \cdots + \frac{Q_m^{N^m}(t)}{\widetilde{\alpha}_m^{N^m}(t)} \\ &= \frac{\alpha_m^1}{\widetilde{\alpha}_m^1(t)} \cdot \frac{Q_m^1(t)}{\alpha_m^1} + \cdots + \frac{\alpha_m^{N^m}}{\widetilde{\alpha}_m^{N^m}(t)} \cdot \frac{Q_m^{N^m}(t)}{\alpha_m^{N^m}},\end{aligned}$$

hence, using (18), for any $t > T_0'$ and any $m \in \mathcal{M}$, we have

$$P\left((1-\epsilon)W_m(t) < \widetilde{W}_m(t) < (1+\epsilon)W_m(t) \,\middle|\, \boldsymbol{Z}(t_0), E \right) = 1, \tag{20}$$

and using (19) and (20), we have

$$P\left(\frac{W_m(t)}{\mu_{i,m}} \geq \overline{W}_i^*(t) > \frac{1}{(1+\epsilon)^2}\widetilde{\overline{W}}_i^*(t)\middle| \boldsymbol{Z}(t_0), E\right) = 1. \quad (21)$$

Using the conditional independence of $\widetilde{\boldsymbol{W}}(t)$ and $\boldsymbol{A}(t)$ from $\boldsymbol{Z}(t_0)$ given $\boldsymbol{Z}(t)$, for any $T > T_0' - t_0$, we have the following for $T_0' \leq t \leq t_0 + T - 1$:

$$\begin{aligned}
&\mathbb{E}\left[\langle \boldsymbol{W}(t), \boldsymbol{A}(t)\rangle | \boldsymbol{Z}(t_0)\right]\\
&\overset{(a)}{=} \mathbb{E}\Bigg[\sum_{m\in\mathcal{M}} W_m(t)\Big(\frac{A_m^1(t)}{\alpha_m^1} + \frac{A_m^2(t)}{\alpha_m^2} + \cdots + \frac{A_m^{N^m}(t)}{\alpha_m^{N^m}}\Big)\\
&\quad \Bigg| \boldsymbol{Z}(t_0)\Bigg]\\
&\overset{(b)}{=} \mathbb{E}\Bigg[\sum_m W_m(t)\Bigg(\frac{1}{\alpha_m^1}\sum_{i\in\mathcal{L}_m^1} A_{i,m}(t) + \frac{1}{\alpha_m^2}\sum_{i\in\mathcal{L}_m^2} A_{i,m}(t)\\
&\quad + \cdots + \frac{1}{\alpha_m^{N^m}}\sum_{i\in\mathcal{L}_m^{N^m}} A_{i,m}(t)\Bigg)\Bigg|\boldsymbol{Z}(t_0)\Bigg]\\
&\overset{(c)}{=} \mathbb{E}\Bigg[\sum_{i\in\mathcal{L}}\sum_{m\in\mathcal{M}}\left(\frac{W_m(t)}{\mu_{i,m}}A_{i,m}(t)\right)\Bigg|\boldsymbol{Z}(t_0)\Bigg]\\
&\overset{(d)}{\leq} \mathbb{E}\Bigg[\sum_{i\in\mathcal{L}}\sum_{m\in\mathcal{M}}\left(\frac{1}{(1-\epsilon)^2}\cdot\frac{\widetilde{W}_m(t)}{\widetilde{\mu}_{i,m}}A_{i,m}(t)\right)\Bigg|\boldsymbol{Z}(t_0), E\Bigg]\\
&\quad +\delta'\cdot\mathbb{E}\Bigg[\sum_{i\in\mathcal{L}}\sum_{m\in\mathcal{M}}\Bigg(\frac{Q_m^1(t)+\cdots+Q_m^{N^m}(t)}{\min\limits_{i,m}\{\mu_{i,m}\}\cdot\min\limits_{i}\{\mu_{i,m}\}}A_{i,m}(t)\Bigg)\\
&\quad \Bigg|\boldsymbol{Z}(t_0), E^c\Bigg]\\
&\overset{(e)}{<} \mathbb{E}\Bigg[\mathbb{E}\Bigg[\sum_{i\in\mathcal{L}}\Bigg(p_e\cdot\frac{1}{(1-\epsilon)^2}\cdot\widetilde{\overline{W}}_i^*(t)A_i(t) + \frac{1-p_e}{(1-\epsilon)^2}\\
&\quad \times\sum_m\frac{\sum_{n=1}^{N^m}Q_m^n(t_0)+N_T(T-t_0)C_A}{\min\limits_{i,m}\{\widetilde{\mu}_{i,m}(t)\}\cdot\min\limits_{i}\{\widetilde{\mu}_{i,m}(t)\}}\cdot C_A\Bigg)\Bigg|\boldsymbol{Z}(t)\Bigg]\\
&\quad \Bigg|\boldsymbol{Z}(t_0), E\Bigg]\\
&\quad +\delta'\cdot\mathbb{E}\Bigg[\sum_{i\in\mathcal{L}}\Bigg(\sum_m\frac{\sum_{n=1}^{N^m}Q_m^n(t_0)+N_T(T-t_0)C_A}{\min_{i,m}\{\mu_{i,m}\}\cdot\min_i\{\mu_{i,m}\}}\cdot C_A\Bigg)\\
&\quad \Bigg|\boldsymbol{Z}(t_0), E^c\Bigg]\\
&\overset{(f)}{<} \frac{1}{(1-\epsilon)^2}\sum_{i\in\mathcal{L}}\mathbb{E}\left[\widetilde{\overline{W}}_i^*(t)\middle|\boldsymbol{Z}(t_0), E\right]\lambda_i + \left(\frac{1}{t^{\delta''}}+\delta'\right)c_0''\\
&\quad \|\boldsymbol{Q}(t_0)\|_1 + c_0',
\end{aligned} \quad (22)$$

where (a) and (b) are simply followed by the definitions of pseudo task arrival process in (10) and $A_m^n(t)$ in (8), respectively. The order of summations is changed in (c). By the law of total probability, (18), and (20), (d) is true, and (e) follows by the routing policy of Blind GB-PANDAS, where an incoming task at the beginning of time slot t is routed to the corresponding sub-queue of the server with the minimum estimated weighted workload with probability $p_e = \max(1-p(t), 0)$ and is routed to the corresponding sub-queue of a server chosen uniformly at random with probability $1-p_e$. Also note that the number of arriving tasks at a time slot is assumed to be upper bounded by C_A. The last step, (f), is true by using (18), upper bounding the exploration probability $1-p_e$ by $\frac{1}{t^{\delta''}}$ given that $\delta'' > 0$ is a constant, and doing simple calculations, where c_0'' and c_0' are constants independent of $\boldsymbol{Z}(t_0)$. Note that minimum value of the estimated service rates, $\min_{i,m}\{\widetilde{\mu}_{i,m}(t)\}$, is lower bounded for any $t \geq t_0$ by a constant which is the minimum of the initialization of service rates and the inverse of the maximum support of CDF functions $F_{i,m}$. We also have

$$\begin{aligned}
&\mathbb{E}\big[\langle \boldsymbol{W}(t), \boldsymbol{w}\rangle | \boldsymbol{Z}(t_0)\big] = \mathbb{E}\left[\sum_{m\in\mathcal{M}} W_m(t)w_m\middle|\boldsymbol{Z}(t_0)\right]\\
&\overset{(a)}{=} \mathbb{E}\left[\sum_{m\in\mathcal{M}}\Big(W_m(t)\sum_{i\in\mathcal{L}}\frac{\lambda_{i,m}}{\mu_{i,m}}\Big)\middle|\boldsymbol{Z}(t_0)\right]\\
&\overset{(b)}{=} \mathbb{E}\left[\sum_{\substack{i\in\mathcal{L}\\ m\in\mathcal{M}}}\frac{W_m(t)}{\mu_{i,m}}\lambda_{i,m}\middle|\boldsymbol{Z}(t_0)\right]\\
&\overset{(c)}{\geq} \sum_{\substack{i\in\mathcal{L}\\ m\in\mathcal{M}}}\frac{1-\delta'}{(1+\epsilon)^2}\mathbb{E}\left[\widetilde{\overline{W}}_i^*(t)\middle|\boldsymbol{Z}(t_0), E\right]\lambda_{i,m}\\
&= \frac{1-\delta'}{(1+\epsilon)^2}\sum_{i\in\mathcal{L}}\mathbb{E}\left[\widetilde{\overline{W}}_i^*(t)\middle|\boldsymbol{Z}(t_0), E\right]\lambda_i,
\end{aligned} \quad (23)$$

where (a) is true by the definition of the ideal workload on a server in (13), note that the ideal workload is not state dependent but $W_m(t)$ is, the order of summations is changed in (b), and (c) is followed by the law of total probability, ignoring the second term, and Equation (21).

Putting (22) and (23) together, for $T > T_0 > T_0'$, we have

$$\begin{aligned}
&\mathbb{E}\Big[\sum_{t=T_0}^{t_0+T-1}\Big(\langle\boldsymbol{W}(t), \boldsymbol{A}(t)\rangle - \langle\boldsymbol{W}(t), \boldsymbol{w}\rangle\Big)\Big|\boldsymbol{Z}(t_0)\Big]\\
&< \sum_{t=T_0}^{t_0+T-1}\Bigg(\left(\frac{1}{(1-\epsilon)^2} - \frac{1-\delta'}{(1+\epsilon)^2}\right)\sum_{i\in\mathcal{L}}\mathbb{E}\left[\widetilde{\overline{W}}_i^*(t)\middle|\boldsymbol{Z}(t_0), E\right]\lambda_i\\
&\quad + \left(\frac{1}{t^{\delta''}}+\delta'\right)c_0''\|\boldsymbol{Q}(t_0)\|_1 + c_0'\Bigg)\\
&\overset{(a)}{<} \frac{16}{9}(4\epsilon+\delta')\cdot\left(\sum_{t=T_0}^{t_0+T-1}\sum_{i\in\mathcal{L}}\mathbb{E}\left[\widetilde{\overline{W}}_i^*(t)\middle|\boldsymbol{Z}(t_0), E\right]\lambda_i\right)\\
&\quad +T\left(\frac{1}{T_0^{\delta''}}+\delta'\right)c_0''\|\boldsymbol{Q}(t_0)\|_1 + Tc_0'\\
&\overset{(b)}{<} \frac{16}{9}(4\epsilon+\delta')\,TN_T\max_i\{\lambda_i\}\\
&\quad \times\left(\mathbb{E}\left[\sum_m\frac{\sum_{n=1}^{N^m}Q_m^1(t_0)+N_T(T-t_0)C_A}{\min_{i,m}\{\widetilde{\mu}_{i,m}(t)\}\cdot\min_i\{\widetilde{\mu}_{i,m}(t)\}}\middle|\boldsymbol{Z}(t_0), E\right]\right)\\
&\quad +T\left(\frac{1}{T_0^{\delta''}}+\delta'\right)c_0''\|\boldsymbol{Q}(t_0)\|_1 + Tc_0'\\
&\overset{(c)}{<} \left(\epsilon+\delta'+\frac{1}{T_0^{\delta''}}\right)Tc_1\|\boldsymbol{Q}(t_0)\|_1 + c_0 = \theta_0\, T\|\boldsymbol{Q}(t_0)\|_1 + c_0,
\end{aligned}$$

where (a) follows by upper bounding $1-\epsilon$, $\frac{1}{(1-\epsilon)^2(1+\epsilon)^2}$, and $\frac{1}{t^{\delta''}}$ by 1, $\frac{16}{9}$, and $\frac{1}{T_0^{\delta''}}$, respectively, and (b) is true by the fact that the number of arriving tasks is bounded by C_A, the number of task types is N_T, and the maximum arrival rate of task types, $\max_i\{\lambda_i\}$, is bounded by the number of servers. Inequality (c) is true by doing simple calculations and using the fact that $\min_{i,m}\{\widetilde{\mu}_{i,m}(t)\}$ is lower bounded by a constant for any $t \geq t_0$ as discussed in (f) of (22).

Remark: θ_0 can be made arbitrary small by choosing ϵ and δ' small and T_0 large enough.

E. Proof of Lemma 5

The proof is similar to the proof of lemma 4 in [11] and is presented for the sake of completeness. By the assumption on boundedness of arrival and service processes, there exists a constant C_A such that for any t_0, t, and T with $t_0 \leq t \leq t_0+T$, we have the following for all $m \in \mathcal{M}$:

$$W_m(t_0) - \frac{T}{\min_n\{\alpha_m^n\}} \leq W_m(t) \leq W_m(t_0) + \frac{TC_A}{\min_n\{\alpha_m^n\}}. \tag{24}$$

On the other hand, by Lemma 2, the ideal workload on a server defined in (13) can be bounded as follows:

$$w_m \leq \frac{1}{1+\delta}, \ \forall m \in \mathcal{M}. \tag{25}$$

Hence,

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\Big(\langle \boldsymbol{W}(t), \boldsymbol{w}\rangle\Big)\Big|\boldsymbol{Z}(t_0)\right] \\
&= \mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\left(\sum_{m=1}^{M} W_m(t)w_m\right)\Big|\boldsymbol{Z}(t_0)\right] \\
&\overset{(a)}{\leq} T\sum_{m=1}^{M}\left(W_m(t_0)w_m + \frac{MT^2C_A}{\min_n\{\alpha_m^n\}}\right) \\
&\overset{(b)}{\leq} \frac{T}{1+\delta}\sum_m W_m(t_0) + \frac{MT^2C_A}{\min_{m,n}\{\alpha_m^n\}},
\end{aligned} \tag{26}$$

where (a) is true by bringing the inner summation on m out of the expectation and using the boundedness property of the workload in Equation (24), and (b) is true by Equation (25).

Before investigating the second term on the left-hand side of Equation (14), $\mathbb{E}\Big[\sum_{t=t_0}^{t_0+T-1}\Big(\langle \boldsymbol{W}(t), \boldsymbol{S}(t)\rangle\Big)\Big|\boldsymbol{Z}(t_0)\Big]$, we propose the following lemma which will be used in lower bounding this second term.

Lemma 8: For any server $m \in \mathcal{M}$ and any t_0, we have the following:

$$\lim_{T\to\infty}\frac{\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\left(\frac{S_m^1(t)}{\alpha_m^1}+\frac{S_m^2(t)}{\alpha_m^2}+\cdots+\frac{S_m^{N^m}(t)}{\alpha_m^{N^m}}\right)\Big|\boldsymbol{Z}(t_0)\right]}{T}=1.$$

We then have the following:

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\Big(\langle \boldsymbol{W}(t), \boldsymbol{S}(t)\rangle\Big)\Big|\boldsymbol{Z}(t_0)\right] \\
&= \mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\sum_{m=1}^{M}\left(W_m(t)\Big(\frac{S_m^1(t)}{\alpha_m^1}+\frac{S_m^2(t)}{\alpha_m^2}\right.\right. \\
&\qquad \left.\left.+\cdots+\frac{S_m^{N^m}(t)}{\alpha_m^{N^m}}\Big)\right)\Big|\boldsymbol{Z}(t_0)\right] \\
&\overset{(a)}{\geq} \sum_{m=1}^{M}\left(W_m(t_0)\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\Big(\frac{S_m^1(t)}{\alpha_m^1}+\frac{S_m^2(t)}{\alpha_m^2}+\right.\right. \\
&\qquad \left.\left.\cdots+\frac{S_m^{N^m}(t)}{\alpha_m^{N^m}}\Big)\Big|\boldsymbol{Z}(t_0)\right]\right) \\
&\quad -\sum_{m=1}^{M}\left(\frac{T}{\min_n\{\alpha_m^n\}}\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\Big(\frac{S_m^1(t)}{\alpha_m^1}+\frac{S_m^2(t)}{\alpha_m^2}\right.\right. \\
&\qquad \left.\left.+\cdots+\frac{S_m^{N^m}(t)}{\alpha_m^{N^m}}\Big)\Big|\boldsymbol{Z}(t_0)\right]\right),
\end{aligned} \tag{27}$$

where (a) follows by bringing the inner summation on m out of the expectation and using the boundedness property of the workload in Equation (24).

Using Lemma 8, for any $0 < \epsilon_0 < \frac{\delta}{1+\delta}$, there exists T_1 such that for any $T \geq T_1$, we have the following for any server $m \in \mathcal{M}$:

$$\begin{aligned}
&1-\epsilon_0 \\
&\leq \frac{\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\left(\frac{S_m^1(t)}{\alpha_m^1}+\frac{S_m^2(t)}{\alpha_m^2}+\cdots+\frac{S_m^{N^m}(t)}{\alpha_m^{N^m}}\right)\Big|\boldsymbol{Z}(t_0)\right]}{T} \\
&\leq 1+\epsilon_0.
\end{aligned}$$

Then continuing on Equation (27), we have the following:

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\Big(\langle \boldsymbol{W}(t), \boldsymbol{S}(t)\rangle\Big)\Big|\boldsymbol{Z}(t_0)\right] \\
&\qquad \geq T(1-\epsilon_0)\sum_{m=1}^{M}W_m(t_0) - \frac{MT^2(1+\epsilon_0)}{\min_{m,n}\{\alpha_m^n\}}.
\end{aligned} \tag{28}$$

Then Lemma 5 is concluded as follows by using equations (26) and (28) and picking $c_1 = \frac{MT^2}{\min_{m,n}\{\alpha_m^n\}}(C_A + 1 + \epsilon_0)$ and $\theta_1 = \frac{1}{\max_{m,n}\{\alpha_m^n\}}\left(\frac{\delta}{1+\delta} - \epsilon_0\right)$, where by our choice of ϵ_0 we have $\theta_1 > 0$:

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\Big(\langle \boldsymbol{W}(t), \boldsymbol{w}\rangle - \langle \boldsymbol{W}(t), \boldsymbol{S}(t)\rangle\Big)\Big|\boldsymbol{Z}(t_0)\right] \\
&\leq -T\left(\frac{\delta}{1+\delta} - \epsilon_0\right)\sum_{m=1}^{M}W_m(t_0) + \frac{MT^2}{\min_{m,n}\{\alpha_m^n\}}(C_A+1+\epsilon_0) \\
&\overset{(a)}{\leq} -\frac{T}{\max_{m,n}\{\alpha_m^n\}}\left(\frac{\delta}{1+\delta} - \epsilon_0\right)\sum_{m=1}^{M}\Big(Q_m^1(t_0)+Q_m^2(t_0) \\
&\qquad +\cdots+Q_m^{N^m}(t_0)\Big) + c_1 \\
&\leq -\theta_1\ T\|\boldsymbol{Q}(t_0)\|_1 + c_1, \ \forall T \geq T_0,
\end{aligned}$$

where (a) is true as $W_m(t_0) \geq \frac{Q_m^1(t_0)+Q_m^2(t_0)+\cdots+Q_m^{N^m}(t_0)}{\max_{m,n}\{\alpha_m^n\}}$.

F. Proof of Lemma 6

This proof is the same as the proof of lemma 5 in [11] and is presented for the sake of completeness. For any server $m \in \mathcal{M}$, let t_m^* be the first time slot after or at time slot t_0 at which the server is available; that is,

$$t_m^* = \min\{\tau : \tau \geq t_0, \Psi_m(\tau) = 0\}, \tag{29}$$

where it is obvious that $\Psi_m(t_m^*) = 0$. Note that for any $t \geq t_0$, we have $\Psi_m(t+1) \leq \Psi_m(t)+1$, which is true by the definition of $\Psi(t)$ that is the number of time slots that server m has spent on the currently in-service task. From time slot t to $t+1$, if a new task comes in service, then $\Psi_m(t+1) = 0$ which results in $\Psi_m(t+1) \le \Psi_m(t)+1$; otherwise, if server m continues giving service to the same task, then $\Psi_m(t+1) = \Psi_m(t)+1$. Thus, if $t_m^* \le t_0+T$, it is easy to find out that $\Psi_m(t_0+T) \le t_0+T-t_m^* \le T$. In the following, we use t_m^* to find a bound on $\mathbb{E}[\Psi_m(t_0+T) - \Psi_m(t_0)|\boldsymbol{Z}(t_0)]$:

$$\begin{aligned}
&\mathbb{E}\Big[\|\boldsymbol{\Psi}(t_0+T)\|_1 - \|\boldsymbol{\Psi}(t_0)\|_1\Big|\boldsymbol{Z}(t_0)\Big]\\
&= \sum_{m=1}^{M} \mathbb{E}\left[\Big(\Psi_m(t_0+T)-\Psi_m(t_0)\Big)\Big|\boldsymbol{Z}(t_0)\right]\\
&= \sum_{m=1}^{M} \bigg\{\mathbb{E}\left[\Big(\Psi_m(t_0+T)-\Psi_m(t_0)\Big)\Big|\boldsymbol{Z}(t_0), t_m^* \le t_0+T\right]\\
&\quad \times P\left(t_m^* \le t_0+T\big|\boldsymbol{Z}(t_0)\right)\\
&\quad +\mathbb{E}\left[\Big(\Psi_m(t_0+T)-\Psi_m(t_0)\Big)\Big|\boldsymbol{Z}(t_0), t_m^* > t_0+T\right]\\
&\quad \times P\left(t_m^* > t_0+T\big|\boldsymbol{Z}(t_0)\right)\bigg\}\\
&\overset{(a)}{\le} \sum_{m=1}^{M} \bigg\{\Big(T-\Psi_m(t_0)\Big) \times P\left(t_m^* > t_0+T\big|\boldsymbol{Z}(t_0)\right)\\
&\quad +T \times P\left(t_m^* > t_0+T\big|\boldsymbol{Z}(t_0)\right)\bigg\}\\
&= -\sum_{m=1}^{M}\bigg(\Psi_m(t_0)\cdot P\left(t_m^* > t_0+T\big|\boldsymbol{Z}(t_0)\right)\bigg) + MT, \quad (30)
\end{aligned}$$

where (a) is true as given that $t_m^* \le t_0+T$ we found that $\Psi_m(t_0+T) \le T$, so $\Psi_m(t_0+T)-\Psi_m(t_0) \le T-\Psi_m(t_0)$, and given that $t_m^* > t_0+T$, it is concluded that server m is giving service to the same task over the whole interval $[t_0, t_0+T]$, which results in $\Psi_m(t_0+T)-\Psi_m(t_0) = T$.

Since the CDF of service time of an n-local task on server m has finite mean, we have the following:

$$\lim_{T\to\infty} P\left(t_m^* \le t_0+T\Big|\boldsymbol{Z}(t_0)\right) = 1,\ \forall m \in \mathcal{M},$$

so for any $\theta_2 \in (0,1)$ there exists T_2 such that for any $T \ge T_2$, we have $P\left(t_m^* \le t_0+T\Big|\boldsymbol{Z}(t_0)\right) \ge \theta_2$, for any $m \in \mathcal{M}$, so Equation (30) follows as below which completes the proof:

$$\begin{aligned}
&\mathbb{E}\Big[\|\boldsymbol{\Psi}(t_0+T)\|_1 - \|\boldsymbol{\Psi}(t_0)\|_1\Big|\boldsymbol{Z}(t_0)\Big]\\
&\quad \le -\theta_2\sum_{m=1}^{M}\Psi_m(t_0) + MT = -\theta_2\|\boldsymbol{\Psi}(t_0)\|_1 + MT. \quad (31)
\end{aligned}$$

G. Proof of Lemma 7

$$\begin{aligned}
&\mathbb{E}\Big[V(\boldsymbol{Z}(t_0+T)) - V(\boldsymbol{Z}(t_0))\Big|\boldsymbol{Z}(t_0)\Big]\\
&= \mathbb{E}\Big[\|\boldsymbol{W}(t_0+T)\|^2 - \|\boldsymbol{W}(t_0)\|^2\Big|\boldsymbol{Z}(t_0)\Big]\\
&\quad +\mathbb{E}\Big[\|\boldsymbol{\Psi}(t_0+T)\|_1 - \|\boldsymbol{\Psi}(t_0)\|_1\Big|\boldsymbol{Z}(t_0)\Big]\\
&\overset{(a)}{=} \mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\Big(\|\boldsymbol{W}(t+1)\|^2 - \|\boldsymbol{W}(t)\|^2\Big)\Big|\boldsymbol{Z}(t_0)\right]\\
&\quad +\mathbb{E}\Big[\|\boldsymbol{\Psi}(t_0+T)\|_1 - \|\boldsymbol{\Psi}(t)\|_1\Big|\boldsymbol{Z}(t_0)\Big]\\
&\overset{(b)}{=} \mathbb{E}\Bigg[\sum_{t=t_0}^{t_0+T-1}\Big(\|\boldsymbol{A}(t)-\boldsymbol{S}(t)+\widetilde{\boldsymbol{U}}(t)\|^2\\
&\quad +2\langle\boldsymbol{W}(t), \boldsymbol{A}(t)-\boldsymbol{S}(t)\rangle + 2\langle\boldsymbol{W}(t), \widetilde{\boldsymbol{U}}(t)\rangle\Big)\Big|\boldsymbol{Z}(t_0)\Bigg]\\
&\quad +\mathbb{E}\Big[\|\boldsymbol{\Psi}(t_0+T)\|_1 - \|\boldsymbol{\Psi}(t)\|_1\Big|\boldsymbol{Z}(t_0)\Big]\\
&\overset{(c)}{\le} 2\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\Big(\langle\boldsymbol{W}(t), \boldsymbol{A}(t)-\boldsymbol{S}(t)\rangle\Big)\Big|\boldsymbol{Z}(t_0)\right]\\
&\quad +\mathbb{E}\Big[\|\boldsymbol{\Psi}(t_0+T)\|_1 - \|\boldsymbol{\Psi}(t)\|_1\Big|\boldsymbol{Z}(t_0)\Big] + c_3'\\
&\overset{(d)}{=} 2\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\Big(\langle\boldsymbol{W}(t), \boldsymbol{A}(t)\rangle - \langle\boldsymbol{W}(t), \boldsymbol{w}\rangle\Big)\Big|\boldsymbol{Z}(t_0)\right]\\
&\quad +2\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\Big(\langle\boldsymbol{W}(t), \boldsymbol{w}\rangle - \langle\boldsymbol{W}(t), \boldsymbol{S}(t)\rangle\Big)\Big|\boldsymbol{Z}(t_0)\right]\\
&\quad +\mathbb{E}\Big[\|\boldsymbol{\Psi}(t_0+T)\|_1 - \|\boldsymbol{\Psi}(t)\|_1\Big|\boldsymbol{Z}(t_0)\Big] + c_3', \quad (32)
\end{aligned}$$

where (a) is true by the telescoping series, (b) follows by using (11) to substitute $\boldsymbol{W}(t+1)$, (c) follows by Lemma 3 and the fact that the task arrival is assumed to be bounded and the service and unused service are also bounded as the number of servers are finite, so the pseudo arrival, service, and unused service are also bounded, and therefore there exists a constant c_1 such that $\|\boldsymbol{A}(t)-\boldsymbol{S}(t)+\widetilde{\boldsymbol{U}}(t)\|^2 \le \frac{c_3'}{T}$, and (d) follows by adding and subtracting the intermediary term $\langle\boldsymbol{W}(t), \boldsymbol{w}\rangle$. On the other hand,

$$\begin{aligned}
&2\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\Big(\langle\boldsymbol{W}(t), \boldsymbol{A}(t)\rangle - \langle\boldsymbol{W}(t), \boldsymbol{w}\rangle\Big)\Big|\boldsymbol{Z}(t_0)\right]\\
&\le 2\mathbb{E}\left[\sum_{t=t_0}^{T_0-1}\langle\boldsymbol{W}(t), \boldsymbol{A}(t)\rangle\Big|\boldsymbol{Z}(t_0)\right]\\
&\quad +2\mathbb{E}\left[\sum_{t=T_0}^{t_0+T-1}\Big(\langle\boldsymbol{W}(t), \boldsymbol{A}(t)\rangle - \langle\boldsymbol{W}(t), \boldsymbol{w}\rangle\Big)\Big|\boldsymbol{Z}(t_0)\right]\\
&\overset{(a)}{\le} 2\mathbb{E}\Bigg[\frac{(T_0-t_0)\cdot C_A}{(\min_{m,n}\{\alpha_m^n\})^2}\sum_{m\in\mathcal{M}}\Big(Q_m^1(t_0)+\cdots+Q_m^{N^m}(t_0)\\
&\quad +N^m\cdot C_A\cdot(T_0-t_0)\Big)\Big|\boldsymbol{Z}(t_0)\Bigg]\\
&\quad +2\mathbb{E}\left[\sum_{t=T_0}^{t_0+T-1}\Big(\langle\boldsymbol{W}(t), \boldsymbol{A}(t)\rangle - \langle\boldsymbol{W}(t), \boldsymbol{w}\rangle\Big)\Big|\boldsymbol{Z}(t_0)\right]\\
&\le 2\mathbb{E}\left[\sum_{t=T_0}^{t_0+T-1}\Big(\langle\boldsymbol{W}(t), \boldsymbol{A}(t)\rangle - \langle\boldsymbol{W}(t), \boldsymbol{w}\rangle\Big)\Big|\boldsymbol{Z}(t_0)\right]\\
&\quad +c_2\|\boldsymbol{Q}(t_0)\|_1 + c_3'', \quad (33)
\end{aligned}$$

where (a) is true by the fact that at most C_A tasks arrive at system in each time slot, and by using the definition of pseudo task arrival in (10). Putting (32) and (33) together, Lemma 7 is proved as follows:

$$\begin{aligned}
&\mathbb{E}\Big[V(\boldsymbol{Z}(t_0+T)) - V(\boldsymbol{Z}(t_0))\Big|\boldsymbol{Z}(t_0)\Big]\\
&\le 2\mathbb{E}\left[\sum_{t=T_0}^{t_0+T-1}\Big(\langle\boldsymbol{W}(t), \boldsymbol{A}(t)\rangle - \langle\boldsymbol{W}(t), \boldsymbol{w}\rangle\Big)\Big|\boldsymbol{Z}(t_0)\right]
\end{aligned}$$

$$+2\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\Big(\langle \boldsymbol{W}(t),\boldsymbol{w}\rangle-\langle \boldsymbol{W}(t),\boldsymbol{S}(t)\rangle\Big)\Big|\boldsymbol{Z}(t_0)\right]$$
$$+\mathbb{E}\Big[\|\boldsymbol{\Psi}(t_0+T)\|_1-\|\boldsymbol{\Psi}(t)\|_1\Big|\boldsymbol{Z}(t_0)\Big]+c_2\|\boldsymbol{Q}(t_0)\|_1+c_3,$$

where $c_3=c_3'+c_3''$.

H. Proof of Lemma 8

The proof is similar to the proof of lemma 6 in [11] and is presented for the sake of completeness. Let t_m^* be the first time slot after or at time slot t_0 at which server m becomes idle, and so is available to serve another task (t_m^* is also defined in (29)); that is,

$$t_m^*=\min\{\tau:\tau\ge t_0,\Psi_m(\tau)=0\}, \tag{34}$$

where, as a reminder, $\Psi_m(\tau)$ is the number of time slots that the m-th server has spent on the task that is receiving service from this server at time slot τ.

Denote the CDF of service time of an n-local task on server m by F_m^n that has finite mean $\alpha_m^n<\infty$; therefore, $t_m^*<\infty$. We then have the following by considering the bounded service:

$$\left(\mathbb{E}\left[\sum_{t=t_m^*}^{t_m^*+T-1}\left(\frac{S_m^1(t)}{\alpha_m^1}+\cdots+\frac{S_m^{N^m}(t)}{\alpha_m^{N^m}}\right)\Bigg|\boldsymbol{Z}(t_0)\right]-\frac{t_m^*-t_0}{\alpha_m^{N^m}}+\frac{1}{\alpha_m^1}\right)\Bigg/T$$
$$\le\frac{\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\left(\frac{S_m^1(t)}{\alpha_m^1}+\frac{S_m^2(t)}{\alpha_m^2}+\cdots+\frac{S_m^{N^m}(t)}{\alpha_m^{N^m}}\right)\Big|\boldsymbol{Z}(t_0)\right]}{T}$$
$$\le\left(\mathbb{E}\left[\sum_{t=t_m^*}^{t_m^*+T-1}\left(\frac{S_m^1(t)}{\alpha_m^1}+\cdots+\frac{S_m^{N^m}(t)}{\alpha_m^{N^m}}\right)\Bigg|\boldsymbol{Z}(t_0)\right]+\frac{1}{\alpha_m^{N^m}}\right)\Bigg/T, \tag{35}$$

where by boundedness of t_m^*,α_m^1, and $\alpha_m^{N^m}$, it is obvious that $\lim_{T\to\infty}\frac{-\frac{t_m^*-t_0}{\alpha_m^{N^m}}+\frac{1}{\alpha_m^1}}{T}=0$ and $\lim_{T\to\infty}\frac{\frac{1}{\alpha_m^{N^m}}}{T}=0$. Hence, by taking the limit of the terms in Equation (35) as T goes to infinity, we have the following:

$$\lim_{T\to\infty}\frac{\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\left(\frac{S_m^1(t)}{\alpha_m^1}+\frac{S_m^2(t)}{\alpha_m^2}+\cdots+\frac{S_m^{N^m}(t)}{\alpha_m^{N^m}}\right)\Big|\boldsymbol{Z}(t_0)\right]}{T}$$
$$=\lim_{T\to\infty}\frac{\mathbb{E}\left[\sum_{t=t_m^*}^{t_m^*+T-1}\left(\frac{S_m^1(t)}{\alpha_m^1}+\frac{S_m^2(t)}{\alpha_m^2}+\cdots+\frac{S_m^{N^m}(t)}{\alpha_m^{N^m}}\right)\Big|\boldsymbol{Z}(t_0)\right]}{T}. \tag{36}$$

Considering the service process as a renewal process, given the scheduling decisions at the end of the renewal intervals in $[t_m^*,t_m^*+T-1]$, all holding times for server m to give service to tasks in its sub-queues are independent. We elaborate on this in the following.

We define renewal processes, $N_m^n(t),\ n\in\{1,2,\cdots,N^m\}$, as follows, where t is an integer valued number:

Let $H_m^n(l)$ be the holding time (service time) of the l-th task that is n-local to server m after time slot t_m^* receiving service from server m, and call $\{H_m^n(l),l\ge 1\}$ the holding process of n-local task type, $n\in\{1,2,\cdots,N^m\}$. Then define $J_m^n(l)=\sum_{i=1}^l H_m^n(l)$ for $l\ge 1$, and let $J_m^n(0)=0$. In the renewal process, $J_m^n(l)$ is the l-th jumping time, or the time at which the l-th occurrence happens, and it has the following relation with the renewal process, $N_m^n(t)$:

$$N_m^n(t)=\sum_{l=1}^{\infty}\mathbb{I}_{\{J_m^n(l)\le t\}}=\sup\{l:J_m^n(l)\le t\}.$$

Another way to define $N_m^n(t)$ is as shown in the following algorithm, where by convention, $N_m^n(0)=0$.

```
1: Set $\tau=t_m^*$, $cntr=0$, $N_m^n(t)=0$
2: while $cntr<t$ do
3:   if $\eta_m(\tau)=n$ then
4:     $cntr++$
5:     $N_m^n(t)\mathrel{+}=S_m^n(\tau)$
6:   end if
7:   $\tau++$
8: end while
```

Another renewal process, $N_m(t)$, is defined as

$$N_m(t)=\sum_{u=t_m^*}^{t_m^*+t-1}\Big(\mathbb{I}_{\{S_m^1(u)=1\}}+\mathbb{I}_{\{S_m^2(u)=1\}}+\cdots+\mathbb{I}_{\{S_m^{N^m}(u)=1\}}\Big).$$

Similarly, let $H_m(l)$ be the holding time (service time) of the l-th task after time slot t_m^* receiving service from server m, and call $\{H_m(l),l\ge 1\}$ the holding process. Then define $J_m(l)=\sum_{i=1}^l H_m(l)$ for $l\ge 1$, and let $J_m(0)=0$. In the renewal process, $J_m(l)$ is the l-th jumping time, or the time at which the l-th occurrence happens, and it has the following relation with the renewal process, $N_m(t)$:

$$N_m(t)=\sum_{l=1}^{\infty}\mathbb{I}_{\{J_m(l)\le t\}}=\sup\{l:J_m(l)\le t\}.$$

Note that the central scheduler makes scheduling decisions for server m at time slots $\{t_m^*+J_m(l),l\ge 1\}$. We denote these scheduling decisions by $D_m(t_m^*)=\Big(\eta_m(t_m^*+J_m(l)):l\ge 1\Big)$. Consider the time interval $[t_m^*,t_m^*+T-1]$ when T goes to infinity. Define ρ_m^n as the fraction of time that server m is busy giving service to tasks that are n-local to this server, in the mentioned interval. Obviously, $\sum_{n=1}^{N^m}\rho_m^n=1$. Then Equation (36) is followed by

$$\lim_{T\to\infty}\frac{\mathbb{E}\left[\sum_{t=t_m^*}^{t_m^*+T-1}\left(\frac{S_m^1(t)}{\alpha_m^1}+\frac{S_m^2(t)}{\alpha_m^2}+\cdots+\frac{S_m^{N^m}(t)}{\alpha_m^{N^m}}\right)\Big|\boldsymbol{Z}(t_0)\right]}{T}$$
$$=\lim_{T\to\infty}\left\{\mathbb{E}\left[\mathbb{E}\left[\sum_{t=t_m^*}^{t_m^*+T-1}\left(\frac{S_m^1(t)}{\alpha_m^1}+\frac{S_m^2(t)}{\alpha_m^2}+\right.\right.\right.\right.$$
$$\left.\left.\left.\left.\cdots+\frac{S_m^{N^m}(t)}{\alpha_m^{N^m}}\right)\Bigg|D_m(t_m^*),\boldsymbol{Z}(t_0)\right]\Bigg|\boldsymbol{Z}(t_0)\right]\right\}\Bigg/T$$
$$=\sum_{n=1}^{N^m}\lim_{T\to\infty}\left(\mathbb{E}\left[\frac{1}{\alpha_m^n}\mathbb{E}\left[\sum_{t=t_m^*}^{t_m^*+T-1}\Big(S_m^n(t)\Big)\right.\right.\right.$$

$$\left|D_m(t_m^*), \boldsymbol{Z}(t_0)\right]\left|\boldsymbol{Z}(t_0)\right]\Bigg) \Bigg/ T$$

$$= \sum_{n=1}^{N^m} \mathbb{E}\left[\frac{1}{\alpha_m^n} \lim_{T\to\infty} \frac{\mathbb{E}\left[N_m^n\left(\rho_m^n T\right)\middle| D_m(t_m^*), \boldsymbol{Z}(t_0)\right]}{T}\middle|\boldsymbol{Z}(t_0)\right]. \tag{37}$$

Note that given $\{D_m(t_m^*), \boldsymbol{Z}(t_0)\}$, the holding times $\{H_m^n(l), l \geq 1\}$ are independent and identically distributed with CDF F_m^n. If $\rho_m^n = 0$, then we do not have to worry about those tasks that are n-local to server m since they receive service from this server for only a finite number of times in time interval $[t_m^*, t_m^* + T - 1]$ as $T \to \infty$, so

$$\lim_{T\to\infty} \frac{\mathbb{E}\left[N_m^n\left(\rho_m^n T\right)\middle| D_m(t_m^*), \boldsymbol{Z}(t_0)\right]}{T} = 0.$$

But if $\rho_m^n > 0$, we can use the strong law of large numbers for renewal process N_m^n to conclude the following:

$$\lim_{T\to\infty} \frac{\mathbb{E}\left[N_m^n\left(\rho_m^n T\right)\middle| D_m(t_m^*), \boldsymbol{Z}(t_0)\right]}{T} = \rho_m^n \cdot \frac{1}{\mathbb{E}[H_m^n(1)]}, \tag{38}$$

where the holding time (service time) $H_m^n(1)$ has CDF F_m^n with expectation $\frac{1}{\alpha_m^n}$. Combining equations (37) and (38), Lemma 8 is concluded as follows:

$$\lim_{T\to\infty} \frac{\mathbb{E}\left[\sum_{t=t_m^*}^{t_m^*+T-1}\left(\frac{S_m^1(t)}{\alpha_m^1} + \frac{S_m^2(t)}{\alpha_m^2} + \cdots + \frac{S_m^{N^m}(t)}{\alpha_m^{N^m}}\right)\middle|\boldsymbol{Z}(t_0)\right]}{T}$$

$$= \sum_{n=1}^{N^m} \mathbb{E}\left[\frac{1}{\alpha_m^n} \cdot \rho_m^n \cdot \alpha_m^n \middle| \boldsymbol{Z}(t_0)\right] = \sum_{n=1}^{N^m} \rho_m^n = 1. \tag{39}$$

References

[1] D. Padua, *Encyclopedia of Parallel Computing*. New York, NY, USA: Springer, 2011.

[2] J. M. Harrison, "Heavy traffic analysis of a system with parallel servers: Asymptotic optimality of discrete-review policies," *Ann. Appl. Probab.*, vol. 8, no. 3, pp. 822–848, Aug. 1998.

[3] J. M. Harrison and M. J. López, "Heavy traffic resource pooling in parallel-server systems," *Queueing Systems*, vol. 33, no. 4, pp. 339–368, 1999.

[4] S. L. Bell and R. J. Williams, "Dynamic scheduling of a system with two parallel servers in heavy traffic with resource pooling: Asymptotic optimality of a threshold policy," *Ann. Appl. Probab.*, vol. 11, no. 3, pp. 608–649, Aug. 2001.

[5] S. Bell and R. Williams, "Dynamic scheduling of a parallel server system in heavy traffic with complete resource pooling: Asymptotic optimality of a threshold policy," *Electron. J. Probab.*, vol. 10, pp. 1044–1115, 2005.

[6] Q. Xie, A. Yekkehkhany, and Y. Lu, "Scheduling with multi-level data locality: Throughput and heavy-traffic optimality," in *Proc. IEEE 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[7] A. Yekkehkhany, "Near-data scheduling for data centers with multiple levels of data locality," M.S. thesis, Dept. Elect. Comput. Eng., Univ. Illinois Urbana-Champaign, Champaign, IL, USA.

[8] A. L. Stolyar, "Maxweight scheduling in a generalized switch: State space collapse and workload minimization in heavy traffic," *Ann. Appl. Probab.*, vol. 14, no. 1, pp. 1–53, Feb. 2004.

[9] A. Mandelbaum and A. L. Stolyar, "Scheduling flexible servers with convex delay costs: Heavy-traffic optimality of the generalized cμ-rule," *Oper. Res.*, vol. 52, no. 6, pp. 836–855, Dec. 2004.

[10] W. Wang, K. Zhu, L. Ying, J. Tan, and L. Zhang, "Maptask scheduling in Mapreduce with data locality: Throughput and heavy-traffic optimality," *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 190–203, Feb. 2016.

[11] A. Yekkehkhany, A. Hojjati, and M. H. Hajiesmaili, "GB-PANDAS: Throughput and heavy-traffic optimality analysis for affinity scheduling," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 45, no. 2, pp. 2–14, Mar. 2018.

[12] R. Pedarsani, J. Walrand, and Y. Zhong, "Robust scheduling for flexible processing networks," *Adv. Appl. Probab.*, vol. 49, no. 2, pp. 603–628, Jun. 2017.

[13] R. Srikant and L. Ying, *Communication Networks: An Optimization, Control, and Stochastic Networks Perspective*. Cambridge, U.K.: Cambridge Univ. Press, 2013.

[14] T. White, *Hadoop: The Definitive Guide*. Sunnyvale, CA, USA: Yahoo! 2010.

[15] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: Fair scheduling for distributed computing clusters," in *Proc. ACM SIGOPS 22nd Symp. Operating Syst. Princ. (SOSP)*. New York, NY, USA: ACM, 2009, pp. 261–276.

[16] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling," in *Proc. 5th Eur. Conf. Comput. Syst.* New York, NY, USA: ACM, 2010, pp. 265–278.

[17] Q. Xie and Y. Lu, "Priority algorithm for near-data scheduling: Throughput and heavy-traffic optimality," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 963–972.

[18] S. H. Lu and P. R. Kumar, "Distributed scheduling based on due dates and buffer priorities," *IEEE Trans. Autom. Control*, vol. 36, no. 12, pp. 1406–1416, Dec. 1991.

[19] J. G. Dai and G. Weiss, "Stability and instability of fluid models for reentrant lines," *Math. Oper. Res.*, vol. 21, no. 1, pp. 115–134, Feb. 1996.

[20] G. Baharian and T. Tezcan, "Stability analysis of parallel server systems under longest queue first," *Math. Methods Oper. Res.*, vol. 74, no. 2, pp. 257–279, Jul. 2011.

[21] A. Dimakis and J. Walrand, "Sufficient conditions for stability of longest-queue-first scheduling: Second-order properties using fluid limits," *Adv. Appl. Probab.*, vol. 38, no. 2, pp. 505–521, 2006.

[22] R. Pedarsani and J. Walrand, "Stability of multiclass queueing networks under longest-queue and longest-dominating-queue scheduling," *J. Appl. Probab.*, vol. 53, no. 2, pp. 421–433, Jun. 2016.

[23] G. Sanjay, G. Howard, and L. Shun-Tak, "The Google file system," in *Proc. 17th ACM Symp. Operating Syst. Princ.*, 2003, pp. 29–43.

[24] G. Ananthanarayanan *et al.*, "Scarlett: Coping with skewed content popularity in mapreduce clusters," in *Proc. 6th Conf. Comput. Syst. (EuroSys)*. New York, NY, USA: ACM, 2011, pp. 287–300.

[25] C. L. Abad, Y. Lu, and R. H. Campbell, "DARE: Adaptive data replication for efficient cluster scheduling," in *Proc. IEEE Int. Conf. Cluster Comput.*, Sep. 2011, pp. 159–168.

[26] D. Mukherjee, S. C. Borst, J. S. H. van Leeuwaarden, and P. A. Whiting, "Universality of power-of-d load balancing in many-server systems," 2016, *arXiv:1612.00723*. [Online]. Available: http://arxiv.org/abs/1612.00723

[27] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. R. Larus, and A. Greenberg, "Join-idle-queue: A novel load balancing algorithm for dynamically scalable Web services," *Perform. Eval.*, vol. 68, no. 11, pp. 1056–1071, Nov. 2011.