A Throughput Optimal Algorithm for Map Task Scheduling in MapReduce with Data Locality

Weina Wang, Kai Zhu and Lei Ying
Electrical, Computer and Energy Engineering
Arizona State University
Tempe, Arizona 85287
{weina.wang, kzhu17, lei.ying.2}@asu.edu

Jian Tan and Li Zhang
IBM T. J. Watson Research Center
Yorktown Heights, New York, 10598
{tanji, zhangli}@us.ibm.com

ABSTRACT

MapReduce/Hadoop framework has been widely used to process large-scale datasets on computing clusters. Scheduling map tasks to improve data locality is crucial to the performance of MapReduce. Many works have been devoted to increasing data locality for better efficiency. However, to the best of our knowledge, fundamental limits of MapReduce computing clusters with data locality, including the capacity region and throughput optimal algorithms, have not been studied. In this paper, we address these problems from a stochastic network perspective. Our focus is to strike the right balance between data-locality and load-balancing to maximize throughput. We present a new queueing architecture and propose a map task scheduling algorithm constituted by the Join the Shortest Queue policy together with the MaxWeight policy. We identify an outer bound on the capacity region, and then prove that the proposed algorithm can stabilize any arrival rate vector strictly within this outer bound. It shows that the algorithm is throughput optimal and the outer bound coincides with the actual capacity region. The proofs in this paper deal with random processing time with different parameters and nonpreemptive tasks, which differentiate our work from many other works, so the proof technique itself is also a contribution of this paper.

1. INTRODUCTION

Processing large-scale datasets has become an increasingly important and challenging problem as the amount of data created by online social networks, healthcare industry, scientific research, etc., explodes. MapReduce/Hadoop [4, 1] is a simple yet powerful framework for processing large-scale datasets in a distributed and parallel fashion, and has been widely used in practice, including Google, Yahoo!, Facebook, Amazon and IBM.

A production MapReduce cluster may even consist of tens of thousands of machines [2]. The stored data are typically organized on distributed file systems (e.g., Google File System (GFS) [5], Hadoop Distributed File System (HDFS) [9]), which divide a large dataset into data chunks (e.g., 64 MB) and store multiple replicas (by default 3) of each chunk on different machines. A data processing request under the MapReduce framework, called a job, consists of two types of tasks: *map* and *reduce*. A map task reads one data chunk and processes it to produce intermediate results (key-value pairs). Then reduce tasks fetch the intermediate results and carry out further computations to produce the final result. Map and reduce tasks are assigned to the machines in the computing cluster by a master node which keeps track of the status of these tasks to manage the computation process. In assigning map tasks, a critical consideration is to place map tasks on or close to machines that store the input data chunks, a problem called *data locality*.

For each task, we call a machine a *local machine* for the task if the data chunk associated with the task is stored locally, and we call this task a *local task* on the machine; otherwise, the machine is called a *remote machine* for the task and correspondingly this task is called a *remote task* on the machine. *Locality* also refers to the fraction of tasks that run on local machines. Improving locality can reduce both the processing time of map tasks and the network traffic load since fewer map tasks need to fetch data remotely. However, assigning all tasks to local machines may lead to an uneven distribution of tasks among machines, i.e., some machines may be heavily congested while others may be idle. Therefore, we need to strike the right balance between data-locality and load-balancing in MapReduce.

In this paper, we call the algorithm that assigns map tasks to machines a *map-scheduling algorithm* or simply a *scheduling algorithm*. There have been several attempts to increase data locality in MapReduce to improve the system efficiency. For example, the default scheduler of Hadoop takes the location information of data chunks into account and attempt to schedule map tasks as close as possible to their machines [12, 13]. The Fair Scheduler [13] is widely used in practice and it becomes the de facto standard in the Hadoop community. It uses a scheduling algorithm called *delay scheduling*, which delays some tasks for a small amount of time to attain higher locality. These algorithms and some other works are reviewed and discussed in detail in the related work section.

While the data locality issue has received a lot of attention and scheduling algorithms that improve data locality have been proposed in the literature and implemented in practice, to the best of our knowledge, none of the existing works have studied the fundamental limits of MapReduce computing clusters with data locality. Basic questions such as *what is the capacity region of a MapReduce computing cluster with data locality, which scheduling algorithm can achieve the full capacity region*, remain open.

In this paper, we will address these basic questions from a stochastic network perspective. Motivated by the observation that a large portion of jobs are map-intensive, and many of them only require map tasks [7], we focus on map-scheduling algorithms and assume reduce tasks are either not required or not the bottleneck of the job processing. Thus we simply use *task* to refer to *map task* in the rest of this paper. We assume the data have been divided into chunks, and each chunk has three replicas stored at three different machines. The computing cluster is modeled as a time-slotted system, in which jobs consisting of a number of map tasks arrive at the beginning of each time slot according to some stochastic process. Each map task processes one data chunk. Within each time slot, a task is completed with probability α at a local machine, or with probability γ ($\gamma < \alpha$) at a remote machine, i.e., the service times are geometrically distributed with different parameters. Based on this model, we establish the following fundamental results:

- First, we present an outer bound on the capacity region of a MapReduce computing cluster with data locality, where the capacity region consists of all arrival vectors for which there exists a scheduling algorithm that stabilizes the system (stability region in [10]).

- We propose a new queueing architecture with one local queue for each machine, storing local tasks associated with the machine, and a common queue for all machines. Based on this new queueing architecture, we propose a two-stage scheduling algorithm under which a newly arrived task is routed to one of the three queues associated with the three local machines or the common queue using the Join the Shortest Queue (JSQ) policy; and when a machine is idle, it selects a task from the local queue associated with it or the common queue using the MaxWeight policy [11].

- We prove that the joint JSQ and MaxWeight scheduling algorithm is throughput optimal, i.e., it can stabilize any arrival rate vector strictly within the outer bound of the capacity region, which also shows that the outer bound is tight and is the same as the actual capacity region. We remark that existing results on MaxWeight-based scheduling algorithms assume deterministic processing (service) time or geometrically distributed processing time with preemptive tasks. To the best of our knowledge, the stability of MaxWeight scheduling with random processing time and nonpreemptive tasks has not been established before. So the proof technique itself is a novel contribution of this paper, and may be extended to prove the stability of MaxWeight scheduling for other applications, in which the service times are geometrically distributed. We remark that recently in [8], the authors studied MaxWeight scheduling for resource allocation in clouds and independently established a similar result with more general service time distributions.

2. RELATED WORK ON DATA LOCALITY IN MAPREDUCE

The default scheduler of Hadoop is a FIFO scheduler [12], which uses a *locality optimization* [12, 13] for map task scheduling as in Google's MapReduce [4]. The scheduling algorithm works as follows: when a machine is ready to run a new task, the scheduler first selects a job according to a priority list, which is a FIFO queue by default, and then it schedules the map task in the job with data closest to the machine (on the machine if possible, otherwise on the same rack, or finally on a different rack).

Hadoop has alternative schedulers available, among which the Fair Scheduler is the de facto standard [13]. The Fair Scheduler aims at giving every user a fair share of the cluster capacity over time, so it uses the naïve fair sharing algorithm for job selection. Instead of using a FIFO queue as the priority list, the naïve fair sharing algorithm uses a certain form of fairness as priority, and the locality optimization is the same. Since strictly following a priority order during scheduling sometimes forces tasks in the head-of-line job to be scheduled on remote machines, the Fair Scheduler uses a technique called *delay scheduling* [13] to further improve locality. When a machine requests a new task, if the job that should be scheduled next according to fairness does not have available local tasks for this machine, it is temporarily skipped for a small amount of time, letting other jobs launch tasks instead. This technique gives more flexibility to the job selection by taking data locality into consideration on the job-level scheduling, which addresses the head-of-line issue in most cases.

The scheduling algorithm Quincy [6] develops a graph-based framework for cluster scheduling under a fine grain cluster resource-sharing model with locality constraints. It maps the fair scheduling problem to a graph datastructure which encodes the competing demands of data locality and fairness and then makes scheduling decisions by solving the classic min-cost flow problem.

3. SYSTEM MODEL

We consider a discrete-time model for a computing cluster with M machines, indexed $1, \cdots, M$. Jobs come in stochastically and when a job comes in, it brings a random number of map tasks, which need to be served by the machines. We assume that each data chunk is replicated and placed at three different machines. Therefore, each task is associated with three local machines. It takes longer time for a machine to process a task if the required data chunk is not stored locally since the machine needs to retrieve the data first. According to the associated local machines, tasks can be classified into *types* denoted by

$$\vec{L} \in \left\{(m_1, m_2, m_3) \in \{1, 2, \cdots, M\}^3 \mid m_1 < m_2 < m_3\right\},$$

where m_1, m_2, m_3 are the indices of the three local machines. We use the notation $m \in \vec{L}$ to indicate machine m is a local machine for type $\vec{L}$ tasks.

3.1 Arrival and Service

Let $A_{\vec{L}}(t)$ denote the number of type $\vec{L}$ tasks arriving at the beginning of time slot t. We assume that the arrival process is temporally i.i.d. with arrival rate $\lambda_{\vec{L}}$. We further assume that there is a positive probability for $A_{\vec{L}}(t)$ to be zero and the arrival processes are bounded. Let $\lambda = (\lambda_{\vec{L}} : \vec{L} \in \mathcal{L})$ be the arrival rate vector, where $\mathcal{L}$ is the set of task types with arrival rates greater than zero; i.e., $\mathcal{L} = \{\vec{L} \mid \lambda_{\vec{L}} > 0\}$.

At each machine, the service times of tasks follow geometric distributions. The parameter of the geometric distribution is α for a task at a local machine, and γ at a remote machine. The service process of a task can be viewed as a sequence of independent trials with success probability α or γ, and the sequence stops once we get a success, i.e., once the task is finished. In this model, we assume $\alpha > \gamma$, so the

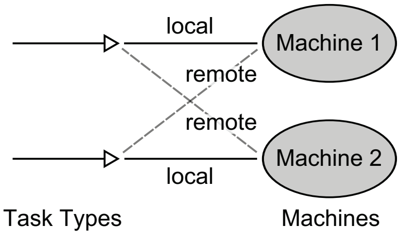


Figure 1: A simple example showing the necessity of task scheduling algorithm under data locality.

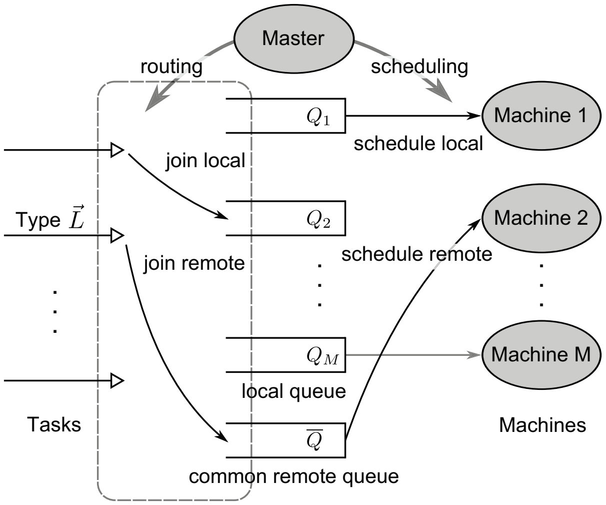


Figure 2: The Queue Architecture and Scheduling Algorithm

average service time of local tasks is less than that of remote tasks, i.e., $1/\alpha < 1/\gamma$. Note that α and γ characterize the different processing efficiency due to data locality.

3.2 Task Scheduling Algorithm

The task scheduling problem is to assign incoming tasks to the machines. Due to data locality, the task scheduling algorithm can significantly affect the efficiency of the system. As a simple example, consider a system with two machines and two task types as in Figure 1. Machine 1 and machine 2 are local machines for the first and the second task type, respectively. If both task types are served by remote machines, then the system can be stable only when the arrival rates of both task types are less than γ, resulting in a throughput less than 2γ. However, if both task types are served by local machines, the system can achieve a throughput close to 2α.

In this paper, we consider a task scheduling algorithm consisting of two parts: routing and scheduling. We present a new queue architecture as illustrated in Figure 2. The master node maintains a queue for each machine m for local tasks, denoted by Q_m and called the *local queue*; and there is a common queue for all machines, denoted by $\overline{Q}$ (or sometimes Q_{M+1}), and called the *common remote queue*. We use a queue length vector

$$Q(t) = \big(Q_1(t), Q_2(t), \cdots, Q_M(t), \overline{Q}(t)\big)$$

to denote the queue lengths at the beginning of time slot t. When a task comes in, the master node routes the task to some queue in the queueing system. When a machine is idle, it picks a task from the corresponding local queue or the common remote queue to serve. These two steps are illustrated in Figure 2. We call the first step *routing*, and with a slight abuse of terminology we call the second step *scheduling*. It should be clear from the context that whether we are referring to the whole task scheduling problem or to this scheduling step. Based on our queue architecture, we propose the following joint routing and scheduling algorithm.

- **Join the Shortest Queue (JSQ) Routing.** When a task comes in, the master node compares the queue lengths of its three local queues and the common remote queue, and then routes the task to the shortest one. Ties are broken randomly. Let $A_{\vec{L},m}(t)$ and $\overline{A}_{\vec{L}}(t)$ denote the arrivals of type $\vec{L}$ tasks allocated to Q_m and $\overline{Q}$, respectively. Then the arrivals allocated to each queue can be expressed by the arrival vector

$$A(t) = \big(A_1(t), \cdots, A_M(t), \overline{A}(t)\big),$$

defined as

$$A_m(t) = \sum_{\vec{L}:\ m \in \vec{L}} A_{\vec{L},m}(t), \quad m = 1, 2, \cdots, M,$$
$$\overline{A}(t) = \sum_{\vec{L} \in \mathcal{L}} \overline{A}_{\vec{L}}(t).$$

- **MaxWeight Scheduling.** If machine m just finished a task at time slot $t-1$, then its working status is *idle*. Otherwise, the machine must be working on some local or remote task. Let $f_m(t) = 0, 1, 2$ denote idle, working on a local task, and working on a remote task, respectively. The working status vector

$$f(t) = (f_1(t), f_2(t), \cdots, f_m(t))$$

and queue length vector $Q(t)$ are reported to the master at the beginning of time slot t, and the master makes scheduling decisions for all the machines based on $f(t)$ and $Q(t)$. The idle machines are scheduled according to the MaxWeight algorithm: suppose machine m is idle at time slot t, then it serves a local task if $\alpha Q_m(t) \geq \gamma \overline{Q}(t)$ and a remote task otherwise. Other machines continue to serve the unfinished tasks, i.e., the execution of tasks is *non-preemptive*. Let $\sigma_m(t)$ denote the scheduling decision of machine m at time slot t, then it is a function of $Q(t)$ and $f_m(t)$, and

$$\sigma_m(t) = \begin{cases} 1 & \text{if a local task is to be served,} \\ 2 & \text{if a remote task is to be served.} \end{cases}$$

Note that $\sigma_m(t)$ indicates which queue machine m is scheduled to serve. It can only take value 1 or 2 since the machine is scheduled to serve either a local task or a remote task. If machine m is not idle, i.e., $f_m(t) = 1$ or 2, the schedule $\sigma_m(t)$ equals to $f_m(t)$ by our settings. However if machine m is idle, i.e., $f_m(t) = 0$, $\sigma_m(t)$ is still either 1 or 2, decided by the master according to the MaxWeight algorithm. We use the schedule vector $\sigma(t) = (\sigma_1(t), \sigma_2(t), \cdots, \sigma_M(t))$ to denote the scheduling decisions of all the machines.

Remark 1. Here we note that the queues in this queue architecture can actually have more structure for fairness consideration. For each queue, instead of maintaining one queue for all the tasks assigned to it, we can divide the queue into multiple subqueues according to the job that the task comes from, i.e., per job subqueues. Then in the scheduling step, after being scheduled to serve some queue, an idle machine can further pick a subqueue to serve for the fairness purpose. However, this change will not affect our analysis throughout this paper, so we only consider this structure in the simulation part.

3.3 Queue Dynamics

In time slot t, first the master checks the working status information $f(t)$ and the queue length $Q(t)$. Then the tasks arrive at the master and the master does the routing and the scheduling, yielding $A(t)$ and $\sigma(t)$. Define

$$\begin{cases} \mu_m^l(t) = \alpha,\ \mu_m^r(t) = 0, & \text{if } \sigma_m(t) = 1, \\ \mu_m^l(t) = 0,\ \mu_m^r(t) = \gamma, & \text{if } \sigma_m(t) = 2. \end{cases}$$

The service from machine m to local queue Q_m and remote queue $\overline{Q}$ are two Bernoulli random variables $S_m^l(t) \sim \text{Bern}\left(\mu_m^l(t)\right)$ and $S_m^r(t) \sim \text{Bern}\left(\mu_m^r(t)\right)$. Hence the service applied to each queue can be expressed by the service vector

$$S(t) = \left(S_1^l(t), \cdots, S_M^l(t), \sum_{m=1}^{M} S_m^r(t) \right),$$

which is the service process we introduced in Section 3.1 with service rate α or γ. Then the queue lengths satisfy the following equations.

- **Local queues.** For any $m = 1, 2, \cdots, M$,

  $$Q_m(t+1) = Q_m(t) + A_m(t) - S_m^l(t) + U_m(t),$$

  where

  $$U_m(t) = \begin{cases} 0 & \text{if } Q_m(t) + A_m(t) \geq 1, \\ S_m^l(t) & \text{if } Q_m(t) + A_m(t) = 0. \end{cases} \tag{1}$$

- **Remote queue.**

  $$\overline{Q}(t+1) = \overline{Q}(t) + \overline{A}(t) - \sum_{m=1}^{M} S_m^r(t) + \overline{U}(t),$$

  where

  $$\overline{U}(t) = \sum_{m=1}^{M} S_m^r(t) - \sum_{m \in \mathcal{A}(t)} S_m^r(t) \tag{2}$$

  and $\mathcal{A}(t)$ is the set of machines which actually have some tasks to serve from the remote queue at time slot t. Note that there can be some machines that attempt to serve the remote queue but fail due to insufficient tasks.

The queue dynamics can thus be expressed as

$$Q(t+1) = Q(t) + A(t) - S(t) + U(t), \tag{3}$$

where $U(t) = \left(U_1(t), U_2(t), \cdots, U_M(t), \overline{U}(t)\right)$ represents the unused service.

In the case that the service time is deterministic or geometrically distributed with preemptive tasks, the queueing process $\{Q(t), t \geq 0\}$ itself is a Markov chain. However, the service times in our model are geometrically distributed with heterogeneous parameters due to data locality, and the tasks are nonpreemptive. Thus we need to also consider the working status vector since $Q(t)$ together with $f(t)$ forms a Markov chain $\{Z(t) = (Q(t), f(t)), t \geq 0\}$. We assume that the initial state of this Markov chain is the zero state, i.e., $Z(t) = (Q(0), f(0)) = \left(0_{(M+1)\times 1}, 0_{M\times 1}\right)$, and the state space $\mathcal{S} \subseteq \mathbb{N}^{M+1} \times \{0,1,2\}^M$ consists of all the states that can be reached from the initial state, where $\mathbb{N}$ is the set of nonnegative integers.

Remark 2. The Markov chain $\{Z(t), t \geq 0\}$ is irreducible and aperiodic under our assumptions. For any state $Z = (Q, f)$ in the state space, since every queue length in the system is finite, the Markov chain will reach the zero state from the state Z within finite time slots if there is no arrivals and all the service is 1 during each time slot, which has a positive probability. Thus there exists $n \in \mathbb{N}$ such that the n-step transition probability from Z to the zero state is positive, i.e., Z can reach the zero state. Therefore, the Markov chain is irreducible. We can also see that the transition probability from the zero state to itself is positive, so the Markov chain is aperiodic.

4. THROUGHPUT OPTIMALITY

In this section, we first identify an outer bound of the capacity region of the system. We then prove that the proposed task scheduling algorithm stabilizes any arrival rate vector strictly within this outer bound, which means that the proposed algorithm is *throughput optimal*, and the capacity region coincides with the outer bound.

4.1 Outer Bound of the Capacity Region

Recall that $\lambda = (\lambda_{\vec{L}} \colon \vec{L} \in \mathcal{L})$ is the arrival rate vector. For any task type $\vec{L} \in \mathcal{L}$, we assume that the arrivals of type $\vec{L}$ tasks that are allocated to machine m has a rate $\lambda_{\vec{L},m}$; then $\lambda_{\vec{L}} = \sum_{m=1}^{M} \lambda_{\vec{L},m}$. The set of rates $\left\{\lambda_{\vec{L},m} \mid \vec{L} \in \mathcal{L}, m = 1, \cdots, M\right\}$ will be called a *decomposition* of the arrival rate vector $\lambda = (\lambda_{\vec{L}} \colon \vec{L} \in \mathcal{L})$ in the rest of this paper, and the index range may be omitted for conciseness. For any machine m, a necessary condition for an arrival rate vector λ to be supportable is that the average arrivals allocated to machine m in one time slot must be served within one time slot, i.e.,

$$\sum_{\vec{L} \colon m \in \vec{L}} \frac{\lambda_{\vec{L},m}}{\alpha} + \sum_{\vec{L} \colon m \notin \vec{L}} \frac{\lambda_{\vec{L},m}}{\gamma} \leq 1, \tag{4}$$

where the left hand side is the time that machine m needs to serve the arrivals allocated to it during one time slot on average, since the service rate is α for local tasks and γ for remote tasks.

Let Λ be the set of arrival rates such that each element has a decomposition satisfying (4). Formally,

$$\begin{aligned} \Lambda = \Bigg\{ \lambda = (\lambda_{\vec{L}} \colon \vec{L} \in \mathcal{L}) \,\Bigg|\, & \lambda_{\vec{L}} = \sum_{m=1}^{M} \lambda_{\vec{L},m},\ \forall \vec{L} \in \mathcal{L}, \\ & \lambda_{\vec{L},m} \geq 0,\ \forall \vec{L} \in \mathcal{L},\ \forall m = 1, \cdots, M, \\ & \sum_{\vec{L} \colon m \in \vec{L}} \frac{\lambda_{\vec{L},m}}{\alpha} + \sum_{\vec{L} \colon m \notin \vec{L}} \frac{\lambda_{\vec{L},m}}{\gamma} \leq 1,\ \forall m = 1, \cdots, M \Bigg\}. \end{aligned} \tag{5}$$

Then Λ gives an outer bound of the capacity region.

4.2 Achievability

Theorem 1 (Throughput Optimality). *The proposed map-scheduling algorithm stabilizes any arrival rate vector strictly within* Λ. *Hence, this algorithm is throughput optimal, and* Λ *is the capacity region of the system.*

Since $\{Z(t) = (Q(t), f(t)), t \geq 0\}$ is an irreducible and aperiodic Markov chain, the stability is defined to be the positive recurrence of this Markov chain. By the extension of the Foster-Lyapunov theorem, to prove the positive recurrence, it is sufficient to find a positive integer T and a Lyapunov function whose T time slot Lyapunov drift is bounded within a finite subset of the state space and negative outside this subset.

Proof. Consider the following Lyapunov function

$$W(Z(t)) = \|Q(t)\|^2 = \sum_{m=1}^{M} Q_m^2(t) + \overline{Q}^2(t).$$

Then for any arrival rate vector $\lambda \in \Lambda^o$, we need to find a finite set $\mathcal{B} \subseteq \mathcal{S}$ and two constants δ and C with $\delta > 0$ such that for some positive integer $T \geq 1$,

$$\mathbb{E}\big[W(Z(t_0+T)) - W(Z(t)) \mid Z(t_0) = Z\big] \leq -\delta \quad \text{if } Z \in \mathcal{B}^c,$$
$$\mathbb{E}\big[W(Z(t_0+T)) - W(Z(t)) \mid Z(t_0) = Z\big] \leq C \quad \text{if } Z \in \mathcal{B}.$$

By the queue dynamics (3), the T time slot Lyapunov drift can be calculated as

$$\begin{aligned} D(Z(t_0)) &= \mathbb{E}\big[W(Z(t_0+T)) - W(Z(t_0)) \mid Z(t_0)\big] \\ &= \mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1} \Big(\|Q(t+1)\|^2 - \|Q(t)\|^2\Big) \,\middle|\, Z(t_0)\right] \\ &= \mathbb{E}\Big[\textstyle\sum_t \big(2\langle Q(t), A(t) - S(t)\rangle + 2\langle Q(t), U(t)\rangle \\ &\qquad + \|A(t) - S(t) + U(t)\|^2\big) \,\Big|\, Z(t_0)\Big]. \end{aligned}$$

When it is clear from the context, the summation range may be omitted. By Lemma 1 in the appendix, the second term $\langle Q(t), U(t)\rangle$ is bounded by a constant. Meanwhile, by our assumption, the arrival vector $A(t)$ and the service $S(t)$ are bounded for any t, and by definition, each entry of the unused service $U(t)$ is no greater than the corresponding entry of $S(t)$, thus also bounded. Therefore, the Lyapunov drift can be bounded as

$$D(Z(t_0)) \leq 2\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1} \langle Q(t), A(t) - S(t)\rangle \,\middle|\, Z(t_0)\right] + B_1, \tag{6}$$

where $B_1 > 0$ is a constant not depending on $Z(t_0)$.

For any arrival rate vector $\lambda \in \mathcal{L} \in \Lambda^o$, since Λ^o is an open set, there exists $\epsilon > 0$ such that $\lambda' = (1+\epsilon)\lambda \in \Lambda$. Suppose the decomposition of λ' which satisfies (4) is $\{\lambda'_{\vec{L},m}\}$. Then

$$\begin{aligned} &\left\{\lambda_{\vec{L},m} \,\middle|\, \vec{L} \in \mathcal{L}, m = 1, 2, \cdots, M\right\} \\ &= \left\{\frac{\lambda'_{\vec{L},m}}{1+\epsilon} \,\middle|\, \vec{L} \in \mathcal{L}, m = 1, 2, \cdots, M\right\} \end{aligned} \tag{7}$$

is a decomposition of λ, and for any m,

$$\sum_{\vec{L}:\, m \in \vec{L}} \frac{\lambda_{\vec{L},m}}{\alpha} + \sum_{\vec{L}:\, m \notin \vec{L}} \frac{\lambda_{\vec{L},m}}{\gamma} \leq \frac{1}{1+\epsilon}. \tag{8}$$

Let $\tilde{\lambda} = \left(\tilde{\lambda}_1, \tilde{\lambda}_2, \cdots, \tilde{\lambda}_{M+1}\right)$ be defined as

$$\begin{aligned} \tilde{\lambda}_m &= \sum_{\vec{L}:\, m \in \vec{L}} \lambda_{\vec{L},m}, m = 1, 2, \cdots, M, \\ \tilde{\lambda}_{M+1} &= \sum_{m=1}^{M} \sum_{\vec{L}:\, m \notin \vec{L}} \lambda_{\vec{L},m}. \end{aligned} \tag{9}$$

Utilizing the term $\langle Q(t), \tilde{\lambda}\rangle$ we can write the expectation in the bound (6) as

$$\begin{aligned} &\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1} \langle Q(t), A(t) - S(t)\rangle \,\middle|\, Z(t_0)\right] \\ &= \mathbb{E}\Big[\textstyle\sum_t \big(\langle Q(t), A(t)\rangle - \langle Q(t), \tilde{\lambda}\rangle\big) \,\Big|\, Z(t_0)\Big] \\ &+ \mathbb{E}\Big[\textstyle\sum_t \big(\langle Q(t), \tilde{\lambda}\rangle - \langle Q(t), S(t)\rangle\big) \,\Big|\, Z(t_0)\Big]. \end{aligned}$$

Since we use the JSQ routing and MaxWeight scheduling algorithm, by Lemma 2 and Lemma 3, when T is large enough,

$$\begin{aligned} &\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1} \langle Q(t), A(t) - S(t)\rangle \,\middle|\, Z(t_0)\right] \\ &\leq -\theta\big(\textstyle\sum_{m=1}^{M} Q_m(t_0) + \overline{Q}(t_0)\big) + B_2, \end{aligned}$$

where $\theta > 0$ and $B_2 > 0$ are two constants not depending on $Z(t_0)$. Hence

$$D(Z(t_0)) \leq -2\theta\big(\textstyle\sum_{m=1}^{M} Q_m(t_0) + \overline{Q}(t_0)\big) + B_1 + 2B_2.$$

Pick any $\delta > 0$ and let

$$\mathcal{B} = \left\{(Q, f) \in \mathcal{S} \,\middle|\, Q_1 + \cdots + Q_M + Q_{M+1} \leq \frac{B_1 + 2B_2 + \delta}{2\theta}\right\}.$$

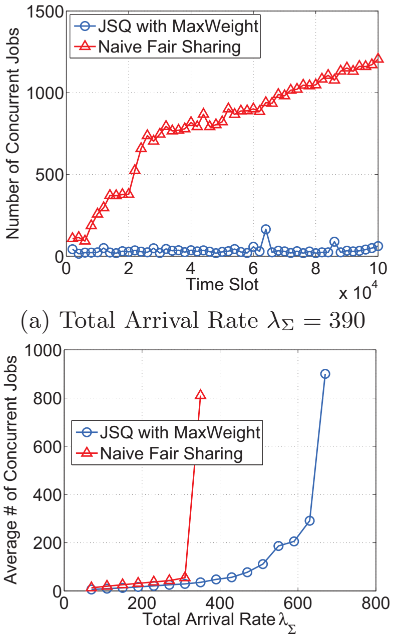
Then $\mathcal{B}$ is a finite subset of $\mathcal{S}$. For any $Z \in \mathcal{B}^c$, $D(Z) \leq -\delta$ and for any $Z \in \mathcal{B}$, $D(Z) \leq B_1 + 2B_2 = B$. This completes the proof of stability. Thus the proposed task scheduling algorithm is throughput optimal, and Λ is the capacity region of the system. □

5. SIMULATIONS

In this section, we use simulations to compare the throughput performance and the delay performance of the proposed algorithm with the naïve fair sharing algorithm in [13]. The related simulation parameters are from mimicking real workload analyzed in [3]. The naïve fair sharing algorithm selects the job with the fewest running tasks and thus satisfies max-min fairness as long as machines become free quickly enough. For the task selection, it greedily searches for a local task in the head-of-line job. The performance of the naïve fair sharing shows a great improvement over the Hadoop's FIFO scheduler according to the evaluations in [13].

5.1 Settings

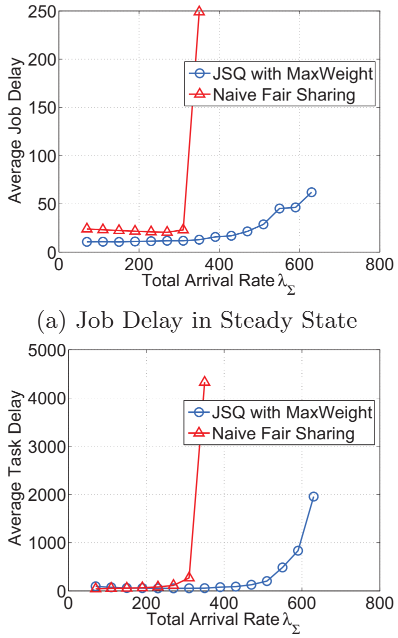
We consider a computing cluster with 1000 machines: 800 machines of them have data on local disks while the other 200 machines do not have data and are just used for computation. A dataset with 10^6 data chunks is maintained on this cluster and each data chunk is replicated and placed at three different machines uniformly. This simulation models the scenario that data chunks constitute a database like the user profile database of Facebook, and each job is some manipulation of the data like searching.



(a) Total Arrival Rate $\lambda_\Sigma = 390$

(b) Throughput Region

Figure 3: Throughput Performance



(a) Job Delay in Steady State

(b) Task Delay in Steady State

Figure 4: Delay Performance.

By the analysis of workloads from [3], the number of tasks in a job follows a power-law distribution, so we use a truncated Pareto distribution ranging from 10 to 100, 000 with shape parameter 1.9 to generate the number of tasks for each job. Each task processes one data chunk uniformly selected from the dataset. The service rates for local tasks and remote tasks are $\alpha = 0.8$ and $\gamma = 0.2$, respectively, so the total task arrival rate λ_Σ should be no larger than $800\alpha + 200\gamma = 680$ per time slot. The number of jobs arriving at each time slot follows a discrete distribution. We run the simulations for the two algorithms for a wide range of total arrival rates to evaluate performance.

As noted in Section 3, we maintain multiple subqueues for each queue, and the subqueue corresponding to the job with the fewest running tasks is selected during scheduling, as in the naïve fair sharing.

5.2 Throughput Performance

We keep track of the number of concurrent jobs in the system to observe stability. Both algorithms can keep the system stable for small arrival rates. Under the naïve fair sharing, the number of concurrent jobs increases almost linearly with the passage of time for $\lambda_\Sigma \geq 350$, which implies instability. The proposed algorithm becomes unstable only at $\lambda_\Sigma = 670$. Figure 3a shows a representative sample of the evolution of the number of concurrent jobs over time, which illustrates the comparison of instability and stability. Figure 3b shows the average number of concurrent jobs in the last 250, 000 time slots for each arrival rate. The turning point positions indicate the difference between the throughput that the two algorithms can achieve. So the throughput under the proposed algorithm is increased by more than 80% compared with the naïve fair sharing.

5.3 Delay Performance

For each total arrival rate λ_Σ, we calculate the average delay for jobs and tasks departing during the last 250, 000 time slots and illustrate the results in Figure 4a and Figure 4b, respectively. We did not plot the results for λ_Σ greater than 390 under the naïve fair sharing and λ_Σ equal to 670 under the proposed algorithm since the delay becomes very large (more than ten times larger) due to instability, which also confirms the throughput difference of the two algorithms. For small arrival rates, the proposed algorithm roughly halves the average job delay compared with the naïve fair sharing (Figure 4a), while the average task delay are roughly the same (Figure 4b).

6. CONCLUSION

We considered map scheduling algorithms in MapReduce with data locality. The primary contribution is the development of a scheduling algorithm which is throughput optimal. We first presented the capacity region of a MapReduce computing cluster with data locality and then we proved the throughput optimality. Simulation results were given not only to illustrate the throughput performance but also to evaluate the delay performance for a large range of total arrival rates.

Acknowledgement

Research supported in part by NSF Grants ECCS-1255425.

7. REFERENCES

[1] Hadoop. `http://hadoop.apache.org`.

[2] G. Ananthanarayanan, S. Agarwal, S. Kandula, A. Greenberg, I. Stoica, D. Harlan, and E. Harris. Scarlett: coping with skewed content popularity in mapreduce clusters. In *Proc. European Conf.*

Computer Systems (EuroSys), pages 287–300, Salzburg, Austria, 2011.

[3] G. Ananthanarayanan, A. Ghodsi, A. Wang, D. Borthakur, S. Kandula, S. Shenker, and I. Stoica. Pacman: coordinated memory caching for parallel jobs. In *Proc. Conf. Networked Systems Design and Implementations (USENIX)*, pages 20–20, 2012.

[4] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *ACM Commun.*, 51(1):107–113, Jan. 2008.

[5] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *Proc. ACM Symp. Operating Systems Principles (SOSP)*, pages 29–43, Bolton Landing, NY, 2003.

[6] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg. Quincy: fair scheduling for distributed computing clusters. In *Proc. ACM Symp. Operating Systems Principles (SOSP)*, pages 261–276, Big Sky, MT, 2009.

[7] S. Kavulya, J. Tan, R. Gandhi, and P. Narasimhan. An analysis of traces from a production mapreduce cluster. In *Proc. IEEE/ACM Int. Conf. Cluster, Cloud and Grid Computing (CCGRID)*, pages 94–103, Melbourne, Australia, 2010.

[8] S. T. Maguluri and R. Srikant. Scheduling jobs with unknown duration in clouds. In *Proc. IEEE Int. Conf. Computer Communications (INFOCOM)*, Turin, Italy, 2013.

[9] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *IEEE Symp. Mass Storage Systems and Technologies (MSST)*, pages 1–10, Incline Villiage, NV, May 2010.

[10] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Trans. Autom. Control*, 4:1936–1948, Dec. 1992.

[11] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Trans. Inf. Theory*, 39:466–478, Mar. 1993.

[12] T. White. *Hadoop: The definitive guide*. Yahoo Press, 2010.

[13] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In *Proc. European Conf. Computer Systems (EuroSys)*, pages 265–278, Paris, France, 2010.

APPENDIX

We follow the same notations as in Section 4.

LEMMA 1 (APPROXIMATE ORTHOGONALITY). *For any time slot t,*

$$\langle Q(t), U(t)\rangle \le M^2,$$

where M is the number of machines in the system.

PROOF. For any time slot t,

$$\langle Q(t), U(t)\rangle = \textstyle\sum_{m=1}^{M} Q_m(t)U_m(t) + \overline{Q}(t)\overline{U}(t).$$

For any $m = 1, \cdots, M$, by the definition of $U_m(t)$ in (1), $U_m(t) > 0$ implies that $Q_m(t) + A_m(t) = 0$, which further implies that $Q_m(t) = 0$ since all the quantities here are nonnegative. Thus $Q_m(t)U_m(t) = 0$.

By the definition of $\overline{U}(t)$ in (2), when there exists some unused service $\overline{U}(t) > 0$, the number of tasks in $\overline{Q}$ must be less than the number of machines which are scheduled to serve $\overline{Q}$, and thus less than M, i.e., $\overline{Q}(t) + \overline{A}(t) < M$. Note that $\overline{U}(t) \le M$, thus $\overline{Q}(t)+\overline{U}(t) \le \overline{Q}(t)+\overline{A}(t)+\overline{U}(t) \le 2M$. Then $\overline{Q}(t)\overline{U}(t) < \left(\overline{Q}(t) + \overline{U}(t)\right)^2/4 \le M^2$. Combining this with the first part yields

$$\langle Q(t), U(t)\rangle = \textstyle\sum_{m=1}^{M} Q_m(t)U_m(t) + \overline{Q}(t)\overline{U}(t) \le M^2.$$

□

LEMMA 2. *Consider an arrival rate vector λ and the corresponding vector $\tilde{\lambda}$ defined in (9). Then under the JSQ routing algorithm, for any t such that $t_0 \le t < t_0 + T$,*

$$\mathbb{E}\big[\langle Q(t), A(t)\rangle - \langle Q(t), \tilde{\lambda}\rangle \;\big|\; Z(t_0)\big] \le 0.$$

PROOF. By taking conditional expectation we have

$$\begin{aligned}
&\mathbb{E}\big[\langle Q(t), A(t)\rangle - \langle Q(t), \tilde{\lambda}\rangle \;\big|\; Z(t_0)\big]\\
&= \mathbb{E}\Big[\mathbb{E}\big[\langle Q(t), A(t)\rangle - \langle Q(t), \tilde{\lambda}\rangle \;\big|\; Z(t)\big] \;\Big|\; Z(t_0)\Big],
\end{aligned} \tag{10}$$

where (10) follows follows from the fact that $Q(t)$ and $A(t)$ are conditionally independent from $Z(t_0)$ given $Z(t)$. Using definitions and changing the order of summations yield

$$\begin{aligned}
&\mathbb{E}\big[\langle Q(t), A(t)\rangle - \langle Q(t), \tilde{\lambda}\rangle \;\big|\; Z(t)\big]\\
&= \mathbb{E}\Big[\textstyle\sum_m Q_m(t)A_m(t) + \overline{Q}(t)\overline{A}(t)\\
&\qquad - \displaystyle\sum_m Q_m(t) \sum_{\vec{L}:\, m\in\vec{L}} \lambda_{\vec{L},m}\\
&\qquad - \overline{Q}(t)\sum_m \sum_{\vec{L}:\, m\notin\vec{L}} \lambda_{\vec{L},m} \;\Big|\; Z(t)\Big]\\
&= \sum_{\vec{L}\in\mathcal{L}}\Bigg(\mathbb{E}\left[\sum_{m\in\vec{L}} Q_m(t)A_{\vec{L},m}(t) + \overline{Q}(t)\overline{A}_{\vec{L}}(t) \;\middle|\; Z(t)\right]\\
&\qquad - \sum_{m\in\vec{L}} Q_m(t)\lambda_{\vec{L},m} - \overline{Q}(t)\sum_{m\notin\vec{L}}\lambda_{\vec{L},m}\Bigg).
\end{aligned} \tag{11}$$

For each type $\vec{L}$, denote

$$Q^*_{\vec{L}}(t) = \min\Big\{\min_{m\in\vec{L}} Q_m(t), \overline{Q}(t)\Big\}.$$

According to the JSQ routing algorithm, $A_{\vec{L}}(t)$ is routed to the shortest one among its three local queues and the common remote queue, so the arrival of type $\vec{L}$ tasks to this queue equals $A_{\vec{L}}(t)$ and the arrivals of type $\vec{L}$ tasks to the other queues are zero. Thus for each $\vec{L}$,

$$\textstyle\sum_{m\in\vec{L}} Q_m(t)A_{\vec{L},m}(t) + \overline{Q}(t)\overline{A}_{\vec{L}}(t) = Q^*_{\vec{L}}(t)A_{\vec{L}}(t),$$

and then

$$\begin{aligned}
&\mathbb{E}\left[\sum_{m\in\vec{L}} Q_m(t)A_{\vec{L},m}(t) + \overline{Q}(t)\overline{A}_{\vec{L}}(t) \;\middle|\; Z(t)\right]\\
&= Q^*_{\vec{L}}(t)\mathbb{E}\big[A_{\vec{L}}(t) \;\big|\; Z(t)\big]\\
&= Q^*_{\vec{L}}(t)\lambda_{\vec{L}}.
\end{aligned}$$

It is obvious that

$$\begin{aligned}&\sum_{m\in\vec{L}} Q_m(t)\lambda_{\vec{L},m} + \overline{Q}(t)\sum_{m\notin\vec{L}}\lambda_{\vec{L},m}\\ &\geq Q^*_{\vec{L}}(t)\left(\sum_{m\in\vec{L}}\lambda_{\vec{L},m}+\sum_{m\notin\vec{L}}\lambda_{\vec{L},m}\right)\\ &= Q^*_{\vec{L}}(t)\lambda_{\vec{L}}.\end{aligned}$$

Therefore by (11), $\mathbb{E}\big[\langle Q(t),A(t)\rangle - \langle Q(t),\tilde{\lambda}\rangle \bigm| Z(t)\big] \leq 0$, and then by (10) we can conclude that

$$\mathbb{E}\big[\langle Q(t),A(t)\rangle - \langle Q(t),\tilde{\lambda}\rangle \bigm| Z(t_0)\big] \leq 0.$$

□

Lemma 3. *Consider an arrival rate vector $\lambda \in \Lambda^o$ and the corresponding vector $\tilde{\lambda}$ defined in* (9). *Then under the MaxWeight scheduling algorithm, there exists a large enough T such that*

$$\begin{aligned}&\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\Big(\langle Q(t),\tilde{\lambda}\rangle - \langle Q(t),S(t)\rangle\Big)\,\middle|\, Z(t_0)\right]\\ &\leq -\theta\left(\sum_{m=1}^{M} Q_m(t_0)+\overline{Q}(t_0)\right)+B_2,\end{aligned}$$

where θ and B_2 are positive constants.

Proof. Consider the following random variables

$$\begin{aligned}&t^*_m = \min\{\tau\colon \tau\geq t_0, f_m(\tau)=0\}, m=1,2,\cdots,M\\ &t^* = \max_{1\leq m\leq M} t^*_m.\end{aligned} \tag{12}$$

Then t^*_m is the first time slot after t_0 at which machine m makes a MaxWeight scheduling decision. We can use t^* to decompose the probability space. Let $T=JK$, where J and K are integers. Then

$$\begin{aligned}&\mathbb{E}\Big[\textstyle\sum_t\big(\langle Q(t),\tilde{\lambda}\rangle - \langle Q(t),S(t)\rangle\big)\Bigm| Z(t_0)\Big]\\ =\ &\mathbb{E}\Big[\textstyle\sum_t\big(\langle Q(t),\tilde{\lambda}\rangle - \langle Q(t),S(t)\rangle\big)\Bigm| Z(t_0), t^*\geq t_0+K\Big]\\ &\cdot\Pr\left(t^*\geq t_0+K \mid Q(t_0),f(t_0)\right) &(13)\\ &+\mathbb{E}\Big[\textstyle\sum_t\big(\langle Q(t),\tilde{\lambda}\rangle - \langle Q(t),S(t)\rangle\big)\Bigm| Z(t_0), t^*< t_0+K\Big]\\ &\cdot\Pr\left(t^*< t_0+K \mid Q(t_0),f(t_0)\right). &(14)\end{aligned}$$

By definitions,

$$\begin{aligned}&\textstyle\sum_t\big(\langle Q(t),\tilde{\lambda}\rangle - \langle Q(t),S(t)\rangle\big)\\ &=\sum_t\sum_m\Bigg(Q_m(t)\sum_{\vec{L}\colon m\in\vec{L}}\lambda_{\vec{L},m}+\overline{Q}(t)\sum_{\vec{L}\colon m\notin\vec{L}}\lambda_{\vec{L},m}\\ &\quad - Q_m(t)S^l_m(t)-\overline{Q}(t)S^r_m(t)\Bigg).\end{aligned}$$

For the term (13), we will show that the conditional expectation is bounded by $TM\big(\sum_m Q_m(t_0)+\overline{Q}(t_0)\big)$ plus some constant. Since the service time follows geometric distribution, it can be seen from Lemma 4 that the probability $\Pr\big(t^*\geq t_0+K \mid Z(t_0)\big)$ can be arbitrarily small when K is large enough. Then the term (13) can be bounded by the product of a small number and the sum of all the queue lengths. Recall that all the arrival processes are bounded and let us assume that they are bounded by $A_{\max}$. Then for any t such that $t_0\leq t<t_0+T$ and any $m=1,2,\cdots,M$,

$$\begin{aligned}Q_m(t)&\leq Q_m(t_0)+(t-t_0)NA_{\max}\leq Q_m(t_0)+TNA_{\max},\\ \overline{Q}(t)&\leq \overline{Q}(t_0)+(t-t_0)NA_{\max}\leq \overline{Q}(t_0)+TNA_{\max}.\end{aligned} \tag{15}$$

Meanwhile, according to (8), we have $\sum_{\vec{L}:m\in\vec{L}}\lambda_{\vec{L},m}<1$ and $\sum_{\vec{L}:m\notin\vec{L}}\lambda_{\vec{L},m}<1$ for any $m=1,2,\cdots,M$. Thus for any t such that $t_0\leq t<t_0+T$,

$$\begin{aligned}&\sum_{m=1}^{M}\Bigg(Q_m(t)\sum_{\vec{L}\colon m\in\vec{L}}\lambda_{\vec{L},m}+\overline{Q}(t)\sum_{\vec{L}\colon m\notin\vec{L}}\lambda_{\vec{L},m}\\ &\qquad - Q_m(t)S^l_m(t)-\overline{Q}(t)S^r_m(t)\Bigg)\\ &\leq \textstyle\sum_{m=1}^{M}\big(Q_m(t_0)+\overline{Q}(t_0)+2TNA_{\max}\big)\\ &\leq M\big(\textstyle\sum_{m=1}^{M}Q_m(t_0)+\overline{Q}(t_0)+2TNA_{\max}\big),\end{aligned}$$

and further

$$\begin{aligned}&\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\Big(\langle Q(t),\tilde{\lambda}\rangle - \langle Q(t),S(t)\rangle\Big)\,\middle|\, Z(t_0),t^*\geq t_0+K\right]\\ &\leq\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}M\left(\sum_{m=1}^{M}Q_m(t_0)+\overline{Q}(t_0)+2TNA_{\max}\right)\right.\\ &\qquad\left.\middle| Z(t_0),t^*\geq t_0+K\right]\\ &=TM\big(\textstyle\sum_{m=1}^{M}Q_m(t_0)+\overline{Q}(t_0)\big)+2T^2MNA_{\max}.\end{aligned} \tag{16}$$

For the term (14), we divide the summation from $t=t_0$ to $t=t_0+T-1$ into two parts: from $t=t_0$ to $t=t^*$ and from $t=t^*+1$ to $t=t_0+T-1$. We will bound the first part by the similar method used for the term (13). Since $t^*<t_0+K$, the bound will be $KM\big(\sum_m Q_m(t_0)+\overline{Q}(t_0)\big)$ plus some constant. The second part will be bounded by a negative quantity proportional to $-(J-1)KM\big(\sum_m Q_m(t_0)+\overline{Q}(t_0)\big)$, and then we can choose large enough J to make the sum negative. Using the properties of conditional expectations we have

$$\begin{aligned}&\mathbb{E}\Big[\textstyle\sum_t\big(\langle Q(t),\tilde{\lambda}\rangle - \langle Q(t),S(t)\rangle\big)\Bigm| Z(t_0),t^*<t_0+K\Big]\\ &=\mathbb{E}\Big[\mathbb{E}\big[\textstyle\sum_t\big(\langle Q(t),\tilde{\lambda}\rangle - \langle Q(t),S(t)\rangle\big)\bigm| t^*,Z(t_0)\big]\\ &\qquad\Bigm| Z(t_0),t^*<t_0+K\Big],\end{aligned} \tag{17}$$

and then

$$\begin{aligned}&\mathbb{E}\Big[\textstyle\sum_t\big(\langle Q(t),\tilde{\lambda}\rangle - \langle Q(t),S(t)\rangle\big)\Bigm| t^*,Z(t_0)\Big]\\ &=\sum_{t=t_0}^{t^*}\Bigg(\sum_{m=1}^{M}\mathbb{E}\Bigg[Q_m(t)\sum_{\vec{L}\colon m\in\vec{L}}\lambda_{\vec{L},m}+\overline{Q}(t)\sum_{\vec{L}\colon m\notin\vec{L}}\lambda_{\vec{L},m}\\ &\qquad - Q_m(t)S^l_m(t)-\overline{Q}(t)S^r_m(t)\ \Bigg|\ t^*,Z(t_0)\Bigg]\Bigg)\\ &\quad+\sum_{t=t^*+1}^{t_0+T-1}\Bigg(\sum_{m=1}^{M}\mathbb{E}\Bigg[Q_m(t)\sum_{\vec{L}\colon m\in\vec{L}}\lambda_{\vec{L},m}+\overline{Q}(t)\sum_{\vec{L}\colon m\notin\vec{L}}\lambda_{\vec{L},m}\\ &\qquad - Q_m(t)S^l_m(t)-\overline{Q}(t)S^r_m(t)\ \Bigg|\ t^*,Z(t_0)\Bigg]\Bigg).\end{aligned} \tag{18}$$

For the summation from $t = t_0$ to $t = t^*$, we still use the bounds in (15), which are based on the boundedness of arrivals, to get

$$\begin{aligned}
&\sum_{t=t_0}^{t^*}\mathbb{E}\Bigg[\sum_{m=1}^{M}\bigg(Q_m(t)\sum_{\vec{L}\colon m\in\vec{L}}\lambda_{\vec{L},m}+\overline{Q}(t)\sum_{\vec{L}\colon m\notin\vec{L}}\lambda_{\vec{L},m}\\
&\qquad - Q_m(t)S^l_m(t)-\overline{Q}(t)S^r_m(t)\bigg)\,\bigg|\, t^*, Z(t_0)\Bigg]\\
&\le \sum_{t=t_0}^{t^*}\mathbb{E}\Bigg[\sum_{m=1}^{M}\left(Q_m(t_0)+\overline{Q}(t_0)+2TNA_{\max}\right)\,\bigg|\, t^*, Z(t_0)\Bigg]\\
&\le (t^*-t_0+1)M\big(\textstyle\sum_{m=1}^{M}Q_m(t_0)+\overline{Q}(t_0)\big)\\
&\quad + 2(t^*-t_0+1)TMNA_{\max}. \qquad (19)
\end{aligned}$$

For the second summation from $t = t^*+1$ to $t = t_0+T-1$, we use the properties of conditional expectations. Since $t^* \le t-1$, t^* is determined by $\big\{Q(t_0), f(t_0), Q(t_0+1), f(t_0+1), \cdots, Q(t-1), f(t-1)\big\}$. Meanwhile, given $Z(t)$, $S(t)$ is independent from all the previous queue lengths and working status. Therefore, for any t such that $t^*+1 \le t \le t_0+T-1$ and any m,

$$\begin{aligned}
&\mathbb{E}\bigg[Q_m(t)\sum_{\vec{L}\colon m\in\vec{L}}\lambda_{\vec{L},m}+\overline{Q}(t)\sum_{\vec{L}\colon m\notin\vec{L}}\lambda_{\vec{L},m}\\
&\qquad - Q_m(t)S^l_m(t)-\overline{Q}(t)S^r_m(t)\,\bigg|\, t^*, Z(t_0)\bigg]\\
&= \mathbb{E}\bigg[\mathbb{E}\bigg[Q_m(t)\sum_{\vec{L}\colon m\in\vec{L}}\lambda_{\vec{L},m}+\overline{Q}(t)\sum_{\vec{L}\colon m\notin\vec{L}}\lambda_{\vec{L},m} \qquad (20)\\
&\qquad -\Big(Q_m(t)S^l_m(t)+\overline{Q}(t)S^r_m(t)\Big)\,\bigg|\, Z(t)\bigg]\,\bigg|\, t^*, Z(t_0)\bigg].\\
&= \mathbb{E}\bigg[Q_m(t)\sum_{\vec{L}\colon m\in\vec{L}}\lambda_{\vec{L},m}+\overline{Q}(t)\sum_{\vec{L}\colon m\notin\vec{L}}\lambda_{\vec{L},m}\\
&\qquad -\Big(Q_m(t)\mathbb{E}\big[S^l_m(t)\mid Z(t)\big]\\
&\qquad\qquad +\overline{Q}(t)\mathbb{E}\big[S^r_m(t)\mid Z(t)\big]\Big)\,\bigg|\, t^*, Z(t_0)\bigg]. \qquad (21)
\end{aligned}$$

The sum of the first two terms satisfies

$$\begin{aligned}
&Q_m(t)\sum_{\vec{L}\colon m\in\vec{L}}\lambda_{\vec{L},m}+\overline{Q}(t)\sum_{\vec{L}\colon m\notin\vec{L}}\lambda_{\vec{L},m}\\
&= \alpha Q_m(t)\sum_{\vec{L}\colon m\in\vec{L}}\frac{\lambda_{\vec{L},m}}{\alpha}+\gamma\overline{Q}(t)\sum_{\vec{L}\colon m\notin\vec{L}}\frac{\lambda_{\vec{L},m}}{\gamma}\\
&\le \max\left\{\alpha Q_m(t), \gamma\overline{Q}(t)\right\}\left(\sum_{\vec{L}\colon m\in\vec{L}}\frac{\lambda_{\vec{L},m}}{\alpha}+\sum_{\vec{L}\colon m\notin\vec{L}}\frac{\lambda_{\vec{L},m}}{\gamma}\right)\\
&\overset{(a)}{\le} \frac{1}{1+\epsilon}\max\left\{\alpha Q_m(t), \gamma\overline{Q}(t)\right\}, \qquad (22)
\end{aligned}$$

where (a) follows from the choice of the decomposition of λ in (8). For the service part in (21), by the construction of $S^l_m(t)$ and $S^r_m(t)$, they are conditionally independent from $Q(t)$ and $f(t)$ given $\sigma_m(t)$. Therefore

$$\begin{aligned}
\mathbb{E}\big[S^l_m(t)\mid Z(t)\big] &= \mathbb{E}\Big[\mathbb{E}\big[S^l_m(t)\mid \sigma_m(t)\big]\,\Big|\, Z(t)\Big]\\
\mathbb{E}\big[S^r_m(t)\mid Z(t)\big] &= \mathbb{E}\Big[\mathbb{E}\big[S^r_m(t)\mid \sigma_m(t)\big]\,\Big|\, Z(t)\Big].
\end{aligned}$$

Consider the following random variables

$$\tau^t_m = \max\left\{\tau\colon \tau\le t, f_m(\tau)=0\right\}, m = 1, 2, \cdots, M. \qquad (23)$$

Then τ^t_m is the last time slot before t at which machine m makes a scheduling decision, so the scheduling decision of machine m remains the same from time slot τ^t_m to t, i.e., $\sigma_m(t) = \sigma_m(\tau^t_m)$. Meanwhile, since the service of each queue in one time slot is also bounded, according to the queue dynamics we have, for any $m = 1, 2, \cdots, M$,

$$\begin{aligned}
Q_m(t) &\ge Q_m(\tau^t_m) - (t-\tau^t_m) \ge Q_m(\tau^t_m) - T,\\
\overline{Q}(t) &\ge \overline{Q}(\tau^t_m) - M(t-\tau^t_m) \ge \overline{Q}(\tau^t_m) - MT.
\end{aligned} \qquad (24)$$

Now the service part can be written as

$$\begin{aligned}
&Q_m(t)\mathbb{E}\big[S^l_m(t)\mid Z(t)\big]+\overline{Q}(t)\mathbb{E}\big[S^r_m(t)\mid Z(t)\big]\\
&= \mathbb{E}\Big[Q_m(t)\mathbb{E}\big[S^l_m(t)\mid \sigma_m(\tau^t_m)\big]\\
&\qquad +\overline{Q}(t)\mathbb{E}\big[S^r_m(t)\mid \sigma_m(\tau^t_m)\big]\,\Big|\, Z(t)\Big]\\
&\ge \mathbb{E}\Big[Q_m(\tau^t_m)\mathbb{E}\big[S^l_m(t)\mid \sigma_m(\tau^t_m)\big]\\
&\qquad +\overline{Q}(\tau^t_m)\mathbb{E}\big[S^r_m(t)\mid \sigma_m(\tau^t_m)\big]\,\Big|\, Z(t)\Big]-(M+1)T\\
&\overset{(a)}{\ge} \mathbb{E}\Big[\max\left\{\alpha Q_m(\tau^t_m), \gamma\overline{Q}(\tau^t_m)\right\}\,\Big|\, Z(t)\Big]-(M+1)T\\
&\overset{(b)}{\ge} \max\left\{\alpha Q_m(t), \gamma\overline{Q}(t)\right\}-(2M+1)T, \qquad (25)
\end{aligned}$$

where the inequality (a) follows from the following facts: if $\sigma_m(\tau^t_m) = 1$, then according to the MaxWeight scheduling we have $\alpha Q_m(\tau^t_m) \ge \gamma\overline{Q}(\tau^t_m)$ and

$$\mathbb{E}\big[S^l_m(t)\mid \sigma_m(\tau^t_m)\big]=\alpha, \quad \mathbb{E}\big[S^r_m(t)\mid \sigma_m(\tau^t_m)\big]=0;$$

otherwise, if $\sigma_m(\tau^t_m) = 2$, then $\alpha Q_m(\tau^t_m) \le \gamma\overline{Q}(\tau^t_m)$ and

$$\mathbb{E}\big[S^l_m(t)\mid \sigma_m(\tau^t_m)\big]=0, \quad \mathbb{E}\big[S^r_m(t)\mid \sigma_m(\tau^t_m)\big]=\gamma.$$

The inequality (b) follows from the boundedness of the service. Combing inequalities (22) and (25) and inserting them back to (21) yield

$$\begin{aligned}
&\mathbb{E}\bigg[Q_m(t)\sum_{\vec{L}\colon m\in\vec{L}}\lambda_{\vec{L},m}+\overline{Q}(t)\sum_{\vec{L}\colon m\notin\vec{L}}\lambda_{\vec{L},m}\\
&\qquad -\Big(Q_m(t)S^l_m(t)+\overline{Q}(t)S^r_m(t)\Big)\,\bigg|\, t^*, Z(t_0)\bigg]\\
&\le \mathbb{E}\bigg[-\frac{\epsilon}{1+\epsilon}\max\{\alpha Q_m(t), \gamma\overline{Q}(t)\}\,\bigg|\, t^*, Z(t_0)\bigg]+(2M+1)T\\
&\le -\frac{\epsilon}{1+\epsilon}\max\left\{\alpha Q_m(t_0), \gamma\overline{Q}(t_0)\right\}+(2M+1)T+\frac{\epsilon TM}{1+\epsilon}\\
&\le -\frac{\epsilon}{1+\epsilon}\left(\frac{\gamma M}{\gamma M+\alpha}\cdot\alpha Q_m(t_0)+\frac{\alpha}{\gamma M+\alpha}\cdot\gamma\overline{Q}(t_0)\right)\\
&\quad +(2M+1)T+\epsilon TM/(1+\epsilon)\\
&= -\frac{\epsilon\alpha\gamma M}{(1+\epsilon)(\gamma M+\alpha)}\left(Q_m(t_0)+\frac{\overline{Q}(t_0)}{M}\right)\\
&\quad +(2M+1)T+\epsilon TM/(1+\epsilon).
\end{aligned}$$

Applying this bound to (20) and (18) gives

$$\begin{aligned}&\sum_{t=t^*+1}^{t_0+T-1}\left(\sum_{m=1}^{M}\mathbb{E}\Bigg[Q_m(t)\sum_{\vec{L}:\,m\in\vec{L}}\lambda_{\vec{L},m}+\overline{Q}(t)\sum_{\vec{L}:\,m\notin\vec{L}}\lambda_{\vec{L},m}\right.\\&\qquad\qquad\left.-\,Q_m(t)S^l_m(t)-\overline{Q}(t)S^r_m(t)\,\Bigg|\,t^*,Z(t_0)\Bigg]\right)\\&\le-\frac{(t_0+T-t^*-1)\epsilon\alpha\gamma M}{(1+\epsilon)(\gamma M+\alpha)}\left(\sum_{m=1}^{M}Q_m(t_0)+\overline{Q}(t_0)\right)\\&\quad+TM\big((2M+1)T+\epsilon TM/(1+\epsilon)\big).\end{aligned}$$

Now we have bounded the two terms which sum up to $\mathbb{E}\Big[\sum_t\big(\langle Q(t),\tilde{\lambda}\rangle-\langle Q(t),S(t)\rangle\big)\,\Big|\,t^*,Z(t_0)\Big]$. Thus by (17)

$$\begin{aligned}&\mathbb{E}\Big[\sum_t\big(\langle Q(t),\tilde{\lambda}\rangle-\langle Q(t),S(t)\rangle\big)\,\Big|\,Z(t_0),t^*<t_0+K\Big]\\&\le\mathbb{E}\Bigg[\left((t^*-t_0+1)M-\frac{(t_0+T-t^*-1)\,\epsilon\alpha\gamma M}{(1+\epsilon)(\gamma M+\alpha)}\right)\\&\qquad\cdot\left(\sum_{m=1}^{M}Q_m(t_0)+\overline{Q}(t_0)\right)\,\Bigg|\,Z(t_0),t^*<t_0+K\Bigg]\\&\quad+\mathbb{E}\Big[2(t^*-t_0+1)TMNA_{\max}+TM\big((2M+1)T\\&\qquad+\epsilon TM/(1+\epsilon)\big)\,\Big|\,Z(t_0),t^*<t_0+K\Big]\\&\le\left(KM-\frac{(J-1)KM\epsilon\alpha\gamma}{(1+\epsilon)(\gamma M+\alpha)}\right)\left(\sum_{m=1}^{M}Q_m(t_0)+\overline{Q}(t_0)\right)\\&\quad+\left(2KTMNA_{\max}+TM\left((2M+1)T+\frac{\epsilon TM}{1+\epsilon}\right)\right).\end{aligned}\tag{26}$$

Let $\phi=\frac{\epsilon\alpha\gamma}{(1+\epsilon)(\gamma M+\alpha)}$ and $J_1=1+\frac{1}{\phi}$. Then we pick a $J>J_1$ to make $KM-(J-1)KM\phi$ negative.

Applying the bounds (16), (26) and the bounds for the probabilities in Lemma 4 to (13) and (14) we obtain,

$$\begin{aligned}&\mathbb{E}\Big[\sum_t\big(\langle Q(t),\tilde{\lambda}\rangle-\langle Q(t),S(t)\rangle\big)\,\Big|\,Z(t_0)\Big]\\&\le TM\left(\sum_{m=1}^{M}Q_m(t_0)+\overline{Q}(t_0)\right)\left(1-\Big(1-(1-\gamma)^K\Big)^M\right)\\&\quad+\left(KM-\frac{(J-1)KM\epsilon\alpha\gamma}{(1+\epsilon)(\gamma M+\alpha)}\right)\left(\sum_{m=1}^{M}Q_m(t_0)+\overline{Q}(t_0)\right)\\&\quad\cdot\Big(1-(1-\gamma)^K\Big)^M+2T^2MNA_{\max}\\&\quad+\left(2KTMNA_{\max}+TM\left((2M+1)T+\frac{\epsilon TM}{1+\epsilon}\right)\right)\\&=TM\big(P_1(K)+(1+\phi)\,P_2(K)/J-\phi P_2(K)\big)\\&\quad\cdot\big(\textstyle\sum_{m=1}^{M}Q_m(t_0)+\overline{Q}(t_0)\big)+B_2,\end{aligned}$$

where

$$P_1(K)=1-\big(1-(1-\gamma)^K\big)^M,\,P_2(K)=\big(1-(1-\gamma)^K\big)^M,$$

and

$$\begin{aligned}B_2=2T^2MNA_{\max}+\Big(&2KTMNA_{\max}\\&+TM\big((2M+1)T+\epsilon TM/(1+\epsilon)\big)\Big)>0.\end{aligned}$$

Now we are going to choose K and J large enough to make the coefficient of the sum of the queue lengths strictly negative. Pick any θ such that $0<\theta<\phi$. Since $P_1(K)$ goes to zero as K goes to infinity, there exists K_1 such that for any $K>K_1$, $P_1(K)\le\frac{\phi-\theta}{3TM}$. Since $P_2(K)$ goes to one as K goes to infinity, there exists K_2 such that for any $K>K_2$, $P_2(K)\ge\frac{1}{TM}-\frac{\phi-\theta}{3\phi TM}$. Let $J_2=\frac{3TM(1+\phi)}{\phi-\theta}$, then for any $J>J_2$, $\frac{1}{J}\,(1+\phi)\,P_2(K)<\frac{\phi-\theta}{3TM}$. By the above inequalities, picking a K such that $K>\max\{K_1,K_2\}$ and a J such that $J>\max\{J_1,J_2\}$ yields

$$\begin{aligned}&TM\big(P_1(K)+(1+\phi)P_2(K)/J-\phi P_2(K)\big)\\&\le(\phi-\theta)/3+(\phi-\theta)/3-\phi\big(1-(\phi-\theta)/3\phi\big)\\&=-\theta.\end{aligned}$$

Thus

$$\begin{aligned}&\mathbb{E}\left[\sum_{t=t_0}^{t_0+T-1}\Big(\langle Q(t),\tilde{\lambda}\rangle-\langle Q(t),S(t)\rangle\Big)\,\Bigg|\,Z(t_0)\right]\\&\le-\theta\left(\sum_{m=1}^{M}Q_m(t_0)+\overline{Q}(t_0)\right)+B_2.\end{aligned}$$

□

Lemma 4. *Consider the random variable t^* defined in (12). Then*

$$\Pr(t^*<t_0+K\mid Z(t_0))\ge\big(1-(1-\gamma)^K\big)^M,\tag{27}$$

$$\Pr(t^*\ge t_0+K\mid Z(t_0))\le1-\big(1-(1-\gamma)^K\big)^M.\tag{28}$$

Proof. Given $f(t_0)$, the distributions of all the service time are determined, so under this condition, the random variables $t^*_m, m=1,2,\cdots,M$ defined in (12) are independent from each other. Therefore

$$\begin{aligned}&\Pr(t^*<t_0+K\mid Z(t_0))\\&=\Pr(t^*_1<t_0+K,t^*_2<t_0+K,\cdots,t^*_M<t_0+K\mid Z(t_0))\\&=\textstyle\prod_{m=1}^{M}\Pr(t^*_m<t_0+K\mid Z(t_0))\\&=\big(1-(1-\alpha)^K\big)^{J_1}\big(1-(1-\gamma)^K\big)^{J_2},\end{aligned}$$

where $J_1=\sum_m I(f_m(t)=1)$ and $J_2=\sum_m I(f_m(t)=2)$ and $I(\cdot)$ is the indicator function. Obviously $J_1+J_2\le M$ and $0<1-(1-\gamma)^K<1$. Then since $\alpha\ge\gamma$,

$$\Pr(t^*<t_0+K\mid Z(t_0))\ge\big(1-(1-\gamma)^K\big)^M$$

and

$$\begin{aligned}\Pr(t^*\ge t_0+K\mid Z(t_0))&=1-\Pr(t^*<t_0+K\mid Z(t_0))\\&\le1-\big(1-(1-\gamma)^K\big)^M.\end{aligned}$$

□