Capacity Analysis of Distributed Computing Systems with Multiple Resource Types

Pengchao Han*†, Shiqiang Wang‡, Kin K. Leung†
*College of Computer Science and Engineering, Northeastern University, China
†Department of Electrical and Electronic Engineering, Imperial College London, UK
‡IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
Email: {p.han18, kin.leung}@imperial.ac.uk, wangshiq@us.ibm.com

***Abstract*—In cloud computing systems, computation, communication, and memory resources are distributed across different physical machines and can be used to execute computational tasks requested by different users. It is challenging to characterize the capacity of such a distributed system, because there exist multiple types of resources and the amount of resources required by different tasks is random. In this paper, we define the capacity as the number of tasks that the system can support with a given overload/outage probability. We derive theoretical formulas for the capacity of distributed systems with multiple resource types, where we consider the power of d choices as the task scheduling strategy in the analysis. Our analytical results describe the capacity of distributed computing systems, which can be used for planning purposes or assisting the scheduling and admission decisions of tasks to various resources in the system. Simulation results using both synthetic and real-world data are also presented to validate the capacity bounds.**

***Index Terms*—Capacity, cloud computing, distributed systems, multiple resource types, power of d choices, software defined networking (SDN)**

I. Introduction

Cloud computing allows flexible configuration of high-level services and sharing of computer resources such as computation, memory, and communication among users. These resources as well as databases are often distributed and connected over high-speed networks or the Internet. Big hi-tech companies including Google, Amazon, and IBM have various offers of cloud computing services and platforms, similar to public utilities. From the system perspective, the cloud-computing infrastructure corresponds to a distributed computing network with different types of data servers and communication resources inter-connected by communication links. Another related notion is edge computing [1], [2], where processing demands are also handled by resources distributed over the communication network, although the emphasis of edge computing is placed on pushing and carrying out the computation close to the end users (i.e., the edge of the infrastructure).

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. P. Han was mainly supported by the China Scholarship Council for her work at Imperial College London.

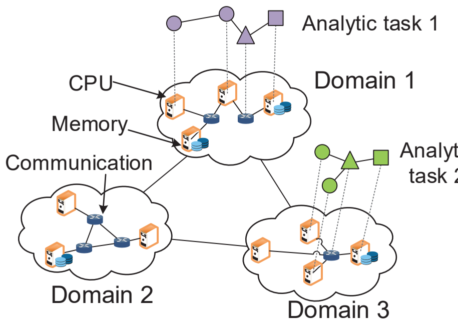

Fig. 1. Distributed computing systems with multiple resource types.

As for communication networks, software defined networking (SDN) [3], [4] has become an emerging architecture with separated data and control planes. An SDN network can consist of multiple interconnected domains (Fig. 1), operated by a single or multiple owners. All control functionalities are implemented on the control plane for operational decisions such as flow-path construction and resource allocation, while the data plane only passively carries out the instructions received from the control plane. Due to the separated control plane, SDN enables programmable network management, easy network reconfiguration and on-demand resource allocation, which can lead to significant improvement of network performance and flexibility.

As pointed out in [5], the notions of cloud/edge computing and SDN are not entirely separate. Indeed, the notion of SDN has been extended beyond communication resources to include computing, storage, data, and other resources to support the computation and communication demands. For example, a set of resources of various types can be identified and designated to support a service (application) or a collection of services. Besides business and civilian applications, the extended notion of SDN with multiple types of resources, referred to as the Software Defined Coalition (SDC) in [6], has also been investigated for defense applications, where a SDC slice consists of a set of computation (servers), communication, and memory resources distributed across multiple domains for executing given analytics tasks, as shown in Fig. 1.

Independent of the civilian or defense nature of the applications, the extended SDN has distributed resources of different types connected by the network. It is important to quantify the "capacity" of a given set (slice) of resources and identify how such distributed resources can be utilized in an optimal way. Toward this goal, we define the *capacity* as the maximum number of analytic tasks that the slice of involved resources can process simultaneously. With the resource capacity known, the system can admit and schedule an appropriate number of tasks for processing. It is challenging to characterize the capacity of resource slice due to random resource requirements of tasks and distributed nature of resources.

In this paper, we derive analytical expressions for the capacity of distributed systems with multiple resource types. Because the exact capacity expression is very difficult to obtain, we present an *achievable lower bound* and an *upper bound* of capacity. Using simulations with both synthetic task arrivals and real-world task arrival traces collected in a data center, we will show that these capacity bounds can closely approximate the actual capacity of the system when parameters in the expression are properly tuned. As the system capacity is related to the way different tasks are assigned to machines, we consider the power of d choices (PODC) as the task assignment strategy in this paper, which is widely used in theoretical analysis and practical systems [7], [8]. The benefit of PODC is that the tradeoff between system capacity and resource control overhead is controllable as preferred, i.e., a larger d gives a larger system capacity but also requires higher control overhead.

The rest of this paper is organized as follows: Section II reviews the related work. The system model and definition of capacity are presented in Section III. The sufficient and necessary conditions for capacity considering PODC assignment are derived in Section IV. The numerical results are presented in Section V, and Section VI concludes this paper.

II. Related Work

The notion of effective capacity is widely explored in wireless communications, which stems from the concept of equivalent bandwidth [9]. In [10], [11], the maximum arrival rate that a wireless link can support with a constrained probability of delay violation is formulated as the effective capacity, based on which the effective capacity of two-hop wireless communication is derived in [12]. Caching is considered for optimizing link capacity in [13], where content replication jointly with routing of user content is optimized to minimize required resources, i.e., maximize link capacity. Compared to the above body of work, task assignment in distributed computing systems is quite different from routing in communication networks, since no pre-defined destination exists for the tasks and the assignment of tasks to machines depends on the task assignment strategy.

The task assignment process can be seen as a variant of the balls-into-bins problem, where M balls (tasks) are allocated into N bins (machines) and intend to achieve load balancing. A general strategy for task assignment is PODC [7], where $d \geq 2$ machines[1] are selected uniformly and randomly from N machines for each task. The least-loaded one among the d selected machines will be chosen to process the task.

[1]PODC can be generalized to $d \geq 1$, we will consider PODC with $d \geq 1$ later in this paper.

The effectiveness of PODC for load balancing with a single resource type and identical tasks is well studied in the literature, such as in [7]. Some work considers the case where tasks have different resource requirements (weights) and the load of a machine is defined as the sum weight of the tasks allocated to it. In [14], the concept of majorisation is used to investigate the relationship among load of machines while assigning weighted tasks using PODC. Markov chain models are used in [15] and [16] to derive the gap between the load of the most loaded machine and the average load. All the above work only considers a single resource type.

For work that considers multiple resource types, machines with different capacities for different types of tasks are considered in [17] and the blocking probability when using PODC is analyzed. The vector scheduling problem is investigated in [18], where overload probability and the load of the most heavily loaded machine are analyzed for PODC strategy. However, only identical tasks with the same resource requirement are considered.

To the best of our knowledge, the scenario where different tasks can require different (random) amounts of resources, in the case with multiple resource types, has not been studied in the literature. The capacity notion in this scenario has not been defined in existing literature either. In this paper, we fill the gap by addressing these problems.

III. System Model and Definitions

We consider a system with multiple types of distributed resources. There are N distributed machines, each has R types of resources (e.g., computation, communication, and memory). The available amount of type-r resource at machine n is normalized to 1 for any n and r. There are T tasks running on the machines in total. The requirement of task t for resource type r is a random variable $X_{t,r} \in [0, 1]$. For each resource type, we assume that the requirements of all tasks for this type of resource (i.e., $X_{t,r}, \forall t$) are independent and identically distributed (i.i.d.), while different distributions may apply to different resource types (i.e., the distribution of $X_{t,r}$ and $X_{t,r'}$ for $r \neq r'$ may be different).

In the system, each task is assigned to one machine that provides resources for this task. Denote $\rho_{n,r}$ as the amount of currently utilized type-r resource at machine n, which is equal to the total requirements for type-r resource of tasks allocated to machine n. To facilitate the capacity analysis later, let $\rho_n := \max_r \rho_{n,r}$ denote the maximum resource utilization among all resources on machine n.

The task allocation follows PODC with $d \geq 1$. When a new task arrives, $d \geq 1$ machines are randomly chosen according to a uniform distribution. The task is assigned to the machine with the minimum ρ_n, among the d selected machines. During the task assignment process, information on resource utilization is exchanged between the d randomly selected machines and a controller, so that the controller can determine which machine is the least-occupied, i.e., has the smallest ρ_n. Different values of d have different control overheads and abilities to balance workload, thus leading to different numbers of tasks that the system can process.

Definition 1 (Capacity)**.** We define the ε-capacity of a distributed computing system as the maximum number of tasks, denoted by M, that the system can serve simultaneously, such that the overload probability is not higher than ε ($\varepsilon > 0$), i.e.,

$$\Pr\left\{\bigcup_{r=1}^{R}\left(\bigcup_{n=1}^{N}\left[\rho_{n,r}\geq 1\right]\right)\right\}\leq\varepsilon. \tag{1}$$

We analyze this capacity in the next section.

IV. Capacity Analysis

The goal of our capacity analysis is to obtain upper bounds of M that serve as sufficient and necessary conditions of (1). The sufficient and necessary conditions give lower and upper bounds of capacity, respectively.

A. Preliminary Lemma

The following lemma is used for the approximation of PODC in the derivation of sufficient and necessary conditions.

Lemma 1. *For vector $\boldsymbol{a} = (a_1, \cdots, a_N)$ with $a_n \geq a_{n+1}$ for $n \in \{1, \cdots, N-1\}$ and probability vector $\mathbf{p} = (p_1, \cdots, p_N)$ with $p_n \leq p_{n+1}$ for $i \in \{1, \cdots, N-1\}$ and $\sum_{n=1}^{N} p_n = 1$, define the weighted average of $\boldsymbol{a}$ as $\sum_{n=1}^{N} p_n a_n$, then*

$$\sum_{n=1}^{N} p_n a_n \leq \bar{p}\sum_{n=1}^{N} a_n,$$

where $\bar{p} = 1/N$ is the mean of all elements in $\mathbf{p}$.

Proof. Assume we have p_i such that $p_i \leq \bar{p} \leq p_{i+1}$, the difference between $\bar{p}\sum_{n=1}^{N} a_n$ and $\sum_{n=1}^{N} p_n a_n$ is

$$\begin{aligned}\bar{p}\sum_{n=1}^{N} a_n - \sum_{n=1}^{N} p_n a_n &= \sum_{n=1}^{i}\left(\bar{p}-p_n\right)a_n - \sum_{n=i+1}^{N}\left(p_n-\bar{p}\right)a_n\\ &\geq \sum_{n=1}^{i}\left(\bar{p}-p_n\right)a_i - \sum_{n=i+1}^{N}\left(p_n-\bar{p}\right)a_i. \end{aligned}\tag{2}$$

Based on the fact that $\sum_{n=1}^{N}\bar{p} = \sum_{n=1}^{N} p_n$, we have

$$\sum_{n=1}^{i}\left(\bar{p}-p_n\right) = \sum_{n=i+1}^{N}\left(p_n-\bar{p}\right). \tag{3}$$

Applying (3) into (2) gives $\bar{p}\sum_{n=1}^{N} a_n - \sum_{n=1}^{N} p_n a_n \geq 0$, and the claim follows. ☐

B. Markov chain

Define vector $\boldsymbol{\rho}(t) := (\rho_n(t), \forall n)$, where $\rho_n(t) := \max_r \rho_{n,r}(t)$ is the maximum utilization among all resource types at machine n, after the t-th task has been assigned. The task assignment process defines a Markov chain over the vector $\boldsymbol{\rho}(t)$ as follows:

1) Choose the machine $i \in \{1, \cdots, N\}$ for the newly arriving t-th task according to PODC.
2) For all $r = 1, 2, ..., R$:
   a) Sample $X_{t+1,r}$ as the requirement of type-r resource of the t-th task, from a given probability distribution.
   b) Set $\rho_{n,r}(t+1) = \rho_{n,r}(t) + X_{t+1,r}$ for $n = i$ (i.e., machine i is chosen for the t-th task) and $\rho_{n,r}(t+1) = \rho_{n,r}(t)$ for $n \neq i$.

We also define a probability vector $\boldsymbol{p} := (p_1, \cdots, p_N)$, where p_i denotes the probability that the new task $t+1$ is assigned to the i-th most loaded machine in terms of $\{\rho_n(t) : \forall n\}$. If we rank $\boldsymbol{\rho}(t)$ in a non-increasing order such that $\rho_1(t) \geq \rho_2(t) \geq \cdots \geq \rho_N(t)$, we have $p_1 \leq p_2 \leq \cdots \leq p_N$ due to the use of PODC strategy. Whenever the context is clear, we write $\boldsymbol{\rho}$, ρ_n and $\rho_{n,r}$ instead of $\boldsymbol{\rho}(t)$, $\rho_n(t)$ and $\rho_{n,r}(t)$.

C. Sufficient Condition for (1)

We first provide the following lemma that serves as an intermediate sufficient condition for further analysis.

Lemma 2. *For any task assignment strategies, if*

$$\sum_{n=1}^{N} E\left(e^{\theta\rho_n}\right) \leq \frac{\varepsilon e^{\theta}}{R} \tag{4}$$

for some $\theta > 0$, then the overload probability less than ε in (1) is guaranteed.

Proof. We note that $E\left(e^{\theta\rho_n}\right)$is the moment generating function (MGF) of ρ_n for $\theta > 0$, Chernoff's bound can be applied. Thus, we have

$$\Pr\left(\rho_n \geq 1\right) \leq \frac{E\left(e^{\theta\rho_n}\right)}{e^{\theta}}, \theta > 0. \tag{5}$$

Substituting (5) into (4) gives

$$R\textstyle\sum_{n=1}^{N}\Pr\left(\rho_n \geq 1\right) \leq \varepsilon.$$

Applying Boole's inequality to the left side of above, we have

$$\begin{aligned}\varepsilon &\geq R\sum_{n=1}^{N}\Pr\left(\rho_n \geq 1\right) \geq \sum_{n=1}^{N}\sum_{r=1}^{R}\Pr\left(\rho_{n,r} \geq 1\right)\\ &\geq \Pr\left\{\bigcup_{r=1}^{R}\left(\bigcup_{n=1}^{N}\left[\rho_{n,r}\geq 1\right]\right)\right\}.\end{aligned}$$

Thus, the overload probability in (1) is confirmed. ☐

Denote $G(\theta) := E\left(e^{\theta\max_r X_{t,r}}\right)$ with $\theta > 0$ for any task t and resource type r as the MGF of $\max_r X_{t,r}$, which can be calculated based on the probability density functions (PDFs) of $X_{t,r}$ for all r (recall that the PDFs of $X_{t,r}$ for different t values are the same, for some given r). Using Lemma 2, we obtain the following sufficient condition for (1).

Theorem 1. *For the PODC strategy with $d \geq 1$, if the number of tasks is less than or equal to*

$$T_l := \frac{\log\frac{\varepsilon}{NR} + \theta}{\log\left(\omega G(\theta) + N - \omega\right) - \log N}, \tag{6}$$

for some $\theta > 0$ and $0 < \omega \leq 1$, then the overload probability in (1) is guaranteed.

Proof. We first define $\Phi(t) := \sum_{n=1}^{N} e^{\theta\rho_n(t)}$ and calculate the mean $\Phi(t)$. Rank $\boldsymbol{\rho}(t)$ in the non-increasing order such that $p_1 \le p_2 \le \cdots \le p_N$ for task $t+1$. The mean increment of $\Phi(t)$ can be calculated as follows:

$$\begin{aligned}
&E\left[\Phi(t+1) - \Phi(t) \,|\, \boldsymbol{\rho}(t)\right] \\
&= E\left[\sum_{n=1}^{N}\left(e^{\theta\rho_n(t+1)} - e^{\theta\rho_n(t)}\right)\middle|\, \boldsymbol{\rho}(t)\right] \\
&\le \sum_{i=1}^{N} p_i E\left[e^{\theta(\rho_i(t)+\max_r X_{t+1,r})} - e^{\theta\rho_i(t)}\middle|\, \boldsymbol{\rho}(t)\right] \qquad (7) \\
&= (G(\theta)-1)\sum_{i=1}^{N} p_i\left(e^{\theta\rho_i(t)}\right) \le \frac{\omega G(\theta) - \omega}{N}\Phi(t).
\end{aligned}$$

In the above, with probability p_i a newly arrived task is allocated to machine i, whose occupancy will increase by $X_{t+1,r}$ for each resource type r. The change of $e^{\theta\rho_n(t)}$ on other machines is 0, i.e., $e^{\theta\rho_n(t+1)} - e^{\theta\rho_n(t)} = 0$ for $n \ne i$, which gives (7), where the inequality is because $\rho_i(t+1) \le \rho_i(t) + \max_r X_{t+1,r}$. Moreover, since $\sum_{i=1}^{N}\left(p_i e^{\theta\rho_i(t)}\right)$ is a weighted average of $e^{\theta\rho_i}$ with higher weights for smaller elements and $\sum_{i=1}^{N} p_i = 1$, we have $\sum_{i=1}^{N}\left(p_i e^{\theta\rho_i(t)}\right) \le \Phi(t)/N$ according to Lemma 1. This gives the last inequality for $\omega = 1$.

Because $E\left[E(X|Y)\right] = E[X]$, from the above we have

$$\begin{aligned}
E\left[\Phi(t+1) - \Phi(t)\right] &= E\left[E\left[\Phi(t+1) - \Phi(t) \,|\, \boldsymbol{\rho}(t)\right]\right] \\
&\le \frac{\omega G(\theta) - \omega}{N} E\left[\Phi(t)\right].
\end{aligned}$$

That is,

$$E\left[\Phi(t+1)\right] \le \frac{\omega G(\theta) + N - \omega}{N} E\left[\Phi(t)\right].$$

For T tasks in total ($T \le T_l$), because $E\left[\Phi(0)\right] = \Phi(0) = \sum_{n=1}^{N} e^{\theta 0} = N$, we have

$$\begin{aligned}
\sum_{n=1}^{N} E\left[e^{\theta\rho_n(T)}\right] &= E\left[\Phi(T_l)\right] \le N\left(\frac{\omega G(\theta) + N - \omega}{N}\right)^{T} \\
&\le N\left(\frac{\omega G(\theta) + N - \omega}{N}\right)^{T_l} = \frac{\varepsilon e^{\theta}}{R}
\end{aligned}$$

for some properly chosen ω, where the first equality is due to the linearity of expectation, the last inequality is because $T \le T_l$ and $\frac{\omega G(\theta)+N-\omega}{N} \ge 1$ for $\omega = 1$, and the last equality is from the definition of T_l in (6). The above is equivalent to (4) in Lemma 2, hence we have proved the theorem. □

D. Necessary Condition for (1)

We first provide the following lemma that serves as an intermediate necessary condition for further analysis.

Lemma 3. *For any task assignment that satisfies (1) with some given ε, we have*

$$\frac{1}{N}\sum_{n=1}^{N} E\left(e^{-\theta\rho_n}\right) \ge \frac{1-\varepsilon}{e^{\theta}} \qquad (8)$$

for any $\theta > 0$.

Proof. A necessary condition for (1) is

$$\begin{aligned}
\frac{1}{N}\sum_{n=1}^{N}\Pr\left(\rho_n \ge 1\right) &\le \max_n \Pr\left(\rho_n \ge 1\right) \qquad (9) \\
&\le \Pr\left\{\bigcup_{r=1}^{R}\left(\bigcup_{n=1}^{N}\left[\rho_{n,r} \ge 1\right]\right)\right\} \le \varepsilon.
\end{aligned}$$

Using Chernoff's bound with $\theta > 0$, we have

$$1 - E\left[e^{-\theta\rho_n}\right]e^{\theta} \le \Pr\left(\rho_n \ge 1\right). \qquad (10)$$

We then apply (10) to the left-hand side of (9). □

Denote $H(\theta) := E\left(e^{-\theta\min_r X_{t,r}}\right)$ with $\theta > 0$ for any task t and resource type r as the MGF (with negative parameter) of $\min_r X_{t,r}$, which can be obtained with the PDFs of $X_{m,r}$ for all r. Note that although we take the minimum of $\{X_{m,r} : \forall r\}$ here, ρ_n is still defined as the maximum of $\{\rho_{n,r} : \forall r\}$ (see Section III). We have the following result.

Theorem 2. *For the PODC strategy with $d \ge 1$ that satisfies (1) with some given ε, the system capacity M satisfies*

$$M \le T_u := \frac{\log(1-\varepsilon) - \theta}{\log\left(vH(\theta) + N - v\right) - \log N} \qquad (11)$$

for any $\theta > 0$ and some $v \ge 1$.

Proof. Define $\Psi_n(t) := e^{-\theta\rho_n(t)}$ for machine n and rank $\boldsymbol{\rho}(t)$ in non-increasing order such that $p_1 \le p_2 \le \cdots \le p_N$ for task $t+1$. The mean increment of $\Psi_n(t)$ for machine n is

$$\begin{aligned}
&E\left[\Psi_n(t+1) - \Psi_n(t) \,|\, \boldsymbol{\rho}(t)\right] \\
&\le p_n e^{-\theta(\rho_n(t)+\min_r X_{t+1,r})} + (1-p_n)e^{-\theta\rho_n(t)} - e^{-\theta\rho_n(t)} \\
&= (H(\theta)-1)\,p_n\Psi_n(t).
\end{aligned}$$

Define $\Psi(t) := \frac{1}{N}\sum_{n=1}^{N}\Psi_n(t)$, we have

$$\begin{aligned}
&E\left[\Psi(t+1) - \Psi(t) \,|\, \boldsymbol{\rho}(t)\right] \\
&= \frac{1}{N}\sum_{n=1}^{N} E\left[\Psi_n(t+1) - \Psi_n(t) \,|\, \boldsymbol{\rho}(t)\right] \\
&\le \frac{1}{N}\sum_{n=1}^{N}(H(\theta)-1)\,p_n\Psi_n(t) \le (H(\theta)-1)\frac{v}{N}\Psi(t),
\end{aligned}$$

where Lemma 1 is used in the last step by considering that $(H(\theta)-1)\,p_n$ decreases with n (note that $H(\theta) - 1 < 0$) and $\Psi_n(t)$ increases with n, hence the result holds for $v = 1$. Using $E\left[E(X|Y)\right] = E[X]$, we have

$$\begin{aligned}
E\left[\Psi(t+1)\right] &= E\left[E\left[\Psi(t+1) - \Psi(t) \,|\, \boldsymbol{\rho}(t)\right]\right] + E\left[\Psi(t)\right] \\
&\le \left((H(\theta)-1)\frac{v}{N} + 1\right)E\left[\Psi(t)\right].
\end{aligned}$$

For M tasks in total, based on the above and using $E\left[\Psi(0)\right] = \sum_{n=1}^{N} e^{-\theta 0}/N = 1$, we have

$$\left(\frac{vH(\theta)+N-v}{N}\right)^{M} \ge E\left[\Psi(M)\right] = \frac{1}{N}\sum_{n=1}^{N} E\left[e^{-\theta\rho_n(M)}\right] \ge \frac{1-\varepsilon}{e^{\theta}}$$

where the equality is due to the linearity of expectation and

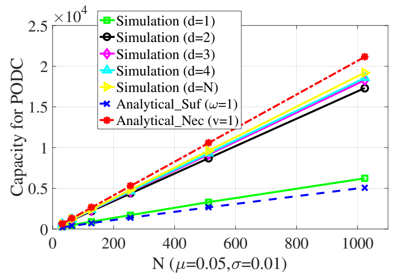

Fig. 2. Performance of capacity bounds for Gaussian resources with $\omega = v = 1$.

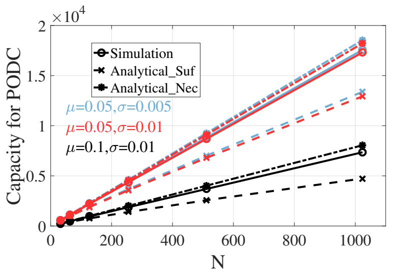

Fig. 3. Comparison of capacity bounds with different μ and σ for a single resource type (i.e., $R = 1$) and tuned ω and v.

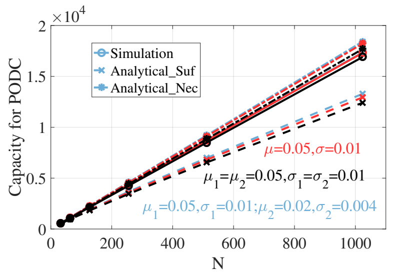

Fig. 4. Comparison of capacity bounds with different R ($R = 1$ for $\mu = 0.05, \sigma = 0.01$ and $R = 2$ elsewhere) and tuned ω and v.

the last inequality is from Lemma 3. Rearranging the above to solve for M proves the theorem. $\square$

E. Discussion

The sufficient condition for (1) given by Theorem 1 represents a lower bound of capacity, because the system is guaranteed to support less than or equal to T_l tasks with an overload probability of ε, where we recall that the capacity is defined as the *maximum* number of tasks the system can support. The necessary condition for (1) given by Theorem 2 gives an upper bound of capacity. Hence, the actual capacity M is bounded by $T_l \leq M \leq T_u$. We note that Theorems 1 and 2 always hold when $\omega = v = 1$, but the parameters ω and v can be tuned to obtain a tighter bound.

V. Numerical Results

We compare our analytical lower and upper bounds of capacity with the actual capacity obtained from simulation, where both Gaussian-distributed and real-world task resource requirements are considered. The MGFs $G(\theta)$ and $H(\theta)$ are computed numerically according to the distribution in each case (the distribution is explained in further details below). The parameter θ in the MGF used in the computation of capacity bounds is also determined numerically via linear search, where we choose θ that gives the largest lower bound and smallest upper bound, so that the bound remains as tight as possible. In all simulations, we fix $\varepsilon = 0.01$.

For simplicity, we use "simulation" to denote the simulated capacity and "analytical" as the results of analytical bounds in the figures presented in the following. We also use "Suf" and "Nec" to denote the sufficient condition (lower bound) and the necessary condition (upper bound) of capacity, respectively.

A. Performance of Gaussian Resource Requirements

We first consider a single resource type where the amount of resource requested by each task follows a Gaussian distribution with parameters μ and σ^2. Fixing $\mu = 0.05$ and $\sigma = 0.01$, Fig. 2 shows the capacity for different d in PODC, when the number of machines varies. We see that $d = N$ performs the best and $d = 2$ outperforms $d = 1$ significantly, which is consistent with known results [7]. Moreover, for this case where $\omega = v = 1$, the theoretical upper and lower capacity bounds hold for all d values with $1 \leq d \leq N$.

The parameters ω and v in the capacity bounds can be tuned so that the theoretical bounds provide a better approximation

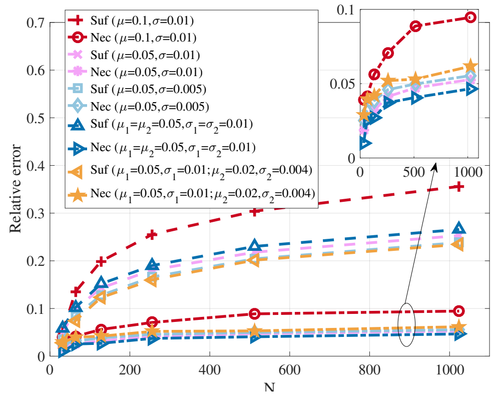

Fig. 5. Relative errors of capacity bounds for Gaussian resource requirements with tuned ω and v.

TABLE I
VALUES OF w AND v FOR GAUSSIAN RESOURCE REQUIREMENTS

Parameter settings	ω	v
$R = 1, \mu = 0.1, \sigma = 0.01$	0.28	1.33
$R = 1, \mu = 0.05, \sigma = 0.01$	0.41	1.16
$R = 1, \mu = 0.05, \sigma = 0.005$	0.41	1.14
$R = 2, \mu_1 = \mu_2 = 0.05, \sigma_1 = \sigma_2 = 0.001$	0.34	1.34
$R = 2, \mu_1 = 0.05, \sigma_1 = 0.01, \mu_2 = 0.01, \sigma_2 = 0.004$	0.28	2.84

for the actual capacity values. For example, we can find the appropriate values of ω and v by minimizing the errors between the simulation results and the analytical bounds when the number of machines $N \in \{20, 40, 60, 80, 100\}$ (a set of settings with small number of machines) . The best ω and v values found with this approach are shown in Table I, for different resource requirement distributions, where we recall that R denotes the number of resource types. We use these tuned ω and v parameters in cases with much larger N in our simulations presented next.

Fig. 3 shows the results with different values of μ and σ for $d = 2$, where a lower value of mean (μ) and standard deviation (σ) in the amount of resource required by each task leads to a larger capacity (i.e., more tasks can be served), as one would intuitively expect. The comparison of single and multiple (two) resource types for $d = 2$ is shown in Fig. 4. We can see that when there are multiple types of resources, the capacity of is dominated by the resource type with larger μ (i.e., the most heavily utilized resource). However, it is uncertain whether multiple resource types will cause higher or lower capacity compared to the case with a single resource

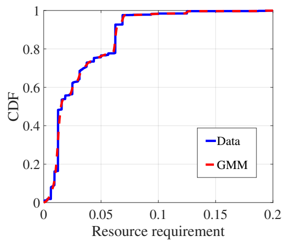

Fig. 6. GMM distribution fitting results for CPU resource.

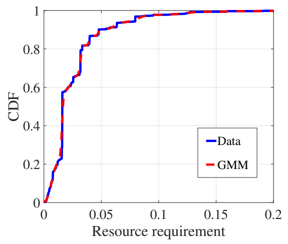

Fig. 7. GMM distribution fitting results for memory resource.

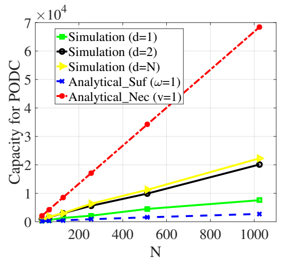

Fig. 8. Performance of capacity bounds for real data with $\omega = v = 1$.

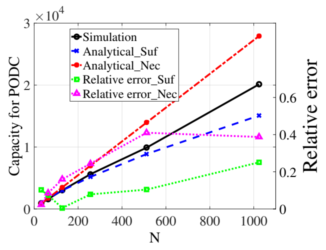

Fig. 9. Performance and relative errors of capacity bounds for real data with tuned parameters ω and v.

type, when the maximum mean value is the same.

We also see that our analytical bounds are close to the simulation results in Figs. 3 and 4. The relative errors (defined as the difference between the simulated capacity and analytical bounds, divided by the simulated capacity), as shown in Fig. 5, are around 0.2 for sufficient conditions (lower bounds) and below 0.1 for necessary conditions (upper bounds). In addition, the analytical bounds can capture the capacity differences when the parameters μ, σ, R, and N are different.

B. Performance of Real-World Resource Requirements

The Google cluster dataset [19] captures the task request dynamics in a real-world computing cluster. It includes the requirements for CPU and memory resources of over $45,000,000$ tasks. To predict the capacity of a distributed system with such task resource requirements, we fit a Gaussian mixture model (GMM) using the dataset and use the MGF of this GMM. Figs. 6 and 7 show the results of GMM fitting for CPU and memory resources respectively. We see that the GMM closely captures the underlying data distribution.

Based on the fitted GMM, Figs. 8 and 9 show the performance of the analytical bounds without or with parameter tuning as well as the relative errors. In the case where ω and v are tuned, their values are $\omega = 0.14, v = 2.42$, where the tuning follows the same approach as in Section V-A. The comparison between different capacity results follow a similar trend as in the Gaussian case in Section V-A.

VI. Conclusion

We have derived theoretical lower and upper bounds of the capacity of distributed computing systems with multiple resources types. The lower and upper bounds correspond to the sufficient and necessary conditions for the overload probability, respectively. The PODC task assignment strategy has been considered, where the trade-off between control overhead and system capacity can be adjusted using the parameter d. The numerical results have shown that our proposed capacity bounds can capture the key characteristics of system capacity and the approximation error is reasonably small. Our results in this paper are useful for describing the capacity of distributed computing systems with multiple resource types, which can be used in system planning, analysis, and resource allocation.

References

[1] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[3] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[4] B. A. A. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[5] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2359–2391, 2017.

[6] "International technology alliance in distributed analytics and information sciences (dais-ita)." https://en.wikipedia.org/wiki/DAIS-ITA.

[7] M. Mitzenmacher, A. W. Richa, and R. Sitaraman, "The power of two random choices: A survey of techniques and results," in *Handbook of Randomized Computing*, pp. 255–312, 2000.

[8] L. Ying, R. Srikant, and X. Kang, "The power of slightly more than one sample in randomized load balancing," *Mathematics of Operations Research*, vol. 42, no. 3, pp. 692–722, 2017.

[9] F. Kelly, "Notes on effective bandwidths," *Stochastic Networks Theory and Applications*, vol. 4, pp. 141–168, 1997.

[10] D. Wu and R. Negi, "Effective capacity: a wireless link model for support of quality of service," *IEEE Transactions on Wireless Communications*, vol. 2, no. 4, pp. 630–643, 2003.

[11] A. Helmy, L. Musavian, and T. Le-Ngoc, "Energy-efficient power adaptation over a frequency-selective fading channel with delay and power constraints," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4529–4541, 2013.

[12] X. Zhang and Q. Zhu, "Statistical QoS provisioning over D2D-offloading based 5G multimedia big-data mobile wireless networks," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 742–747, 2018.

[13] S. Gitzenis, G. Paschos, and L. Tassiulas, "Asymptotic laws for joint content replication and delivery in wireless networks," *CoRR*, vol. abs/1201.3095, pp. 1–16, 2012.

[14] P. Berenbrink, T. Friedetzky, Z. Hu, and R. Martin, "On weighted balls-into-bins games," *Theoretical Computer Science*, vol. 409, no. 3, pp. 511 – 520, 2008.

[15] Y. Peres, K. Talwar, and U. Wieder, "The $(1 + \beta)$-choice process and weighted balls-into-bins," in *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1613–1619, 2010.

[16] U. Wieder, "Hashing, load balancing and multiple choice," *Foundations and Trends in Theoretical Computer Science*, vol. 12, no. 3-4, pp. 275–379, 2017.

[17] Q. Xie, X. Dong, Y. Lu, and R. Srikant, "Power of d choices for large-scale bin packing: A loss model," *SIGMETRICS Perform. Eval. Rev.*, vol. 43, no. 1, pp. 321–334, 2015.

[18] Y. Azar, I. R. Cohen, and D. Panigrahi, "Randomized algorithms for online vector load balancing," in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pp. 980–991, 2018.

[19] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., White Paper*, pp. 1–14, 2011.