Elastic Reliability Optimization Through Peer-to-Peer Checkpointing in Cloud Computing

Juzi Zhao, Yu Xiang, Tian Lan, *Member, IEEE*, H. Howie Huang, *Senior Member, IEEE*, and Suresh Subramaniam, *Fellow, IEEE*

Abstract—Modern day data centers coordinate hundreds of thousands of heterogeneous tasks and aim at delivering highly reliable cloud computing services. Although offering equal reliability to all users benefits everyone at the same time, users may find such an approach either inadequate or too expensive to fit their individual requirements, which may vary dramatically. In this paper, we propose a novel method for providing elastic reliability optimization in cloud computing. Our scheme makes use of peer-to-peer checkpointing and allows user reliability levels to be jointly optimized based on an assessment of their individual requirements and total available resources in the data center. We show that the joint optimization can be efficiently solved by a distributed algorithm using dual decomposition. The solution improves resource utilization and presents an additional source of revenue to data center operators. Our validation results suggest a significant improvement of reliability over existing schemes.

Index Terms—Cloud computing, data center, reliability, checkpoint, optimization

1 Introduction

In today's public clouds, reliability is provided as a fixed service parameter, e.g., Amazon published that its EC2 users can expect 99.95 percent uptime in terms of reliability, which corresponds to a once-a-week failure ratio [1].[1] It is up to the users to harden the tasks running within Virtual Machine (VM) instances to achieve better reliability if so desired.

Clearly, this *all-or-nothing* approach is unsatisfactory—users may find it either inadequate or too expensive to fit their reliability requirements, which have been shown to vary dramatically [2]. Current solutions to achieve high reliability in data centers include VM replication [3], and checkpointing [4], [5], [6], [26], [27]. In particular, several scheduling algorithms for balancing checkpoint workload and reliability have been proposed in [7], [8], [9], with an extension in [10] by considering dynamic VM prices. Nevertheless, previous work has only investigated how to derive optimal checkpoint policies to minimize the execution time of a single task.

In this paper, we propose a novel utility-optimization approach to provide elastic reliability, where flexible service-level agreements (SLAs) are made available to the users based on a joint assessment of their individual reliability requirements and total resources available in the data center.

While providing elastic reliability is undoubtedly appealing to data center operators, it also comes with great technical challenges. To optimize reliability under network resource constraints, data center operators not only have to decide checkpoint scheduling, but also need to determine where to place VM checkpoints, and how to route the checkpoint traffic among peers with sufficient bandwidth. A global checkpoint scheduling (i.e., jointly determining reliability levels and checkpoint time sequences for all users) is preferred because all users share the same pool of resources, which also calls for an adaptive resource allocation scheme in accordance with the user demands [28]. Intuitively, users with higher demands and budgets should be assigned more resources, resulting in better reliability. Their checkpoint events should also be coordinated to mitigate resource interference among themselves and with existing tasks. In this paper, we model different reliability requirements by user-specific utilities, which are increasing functions of reliability (i.e., service uptime). Therefore, the problem of joint reliability maximization can be formulated as an optimization, in which data center operators need to find checkpoint scheduling and make routing/placement decisions in order to maximize an aggregate utility of reliability.

This paper harnesses checkpointing technique with utility optimization to provide joint reliability maximization under resource constraints in data centers. A main feature of our approach is a peer-to-peer checkpointing mechanism, which enables VM images to be transferred and saved among neighboring peers, eliminating the need for any central storage where network congestion gets magnified across all hosts and VMs. It is demonstrated that such a distributed approach is effective to make faster checkpoints and recovery [7]. For data center operators, it also presents an additional source of revenue by exploiting under-utilized resources. For example, at any time only a few core

1. We follow some of the literature (e.g., [24], [25]), where "reliability" is used to denote the ratio of total uptime (operational time) to the total service time.

- J. Zhao is with the Department of Electrical and Computer Engineering, University of Massachusetts-Lowell, Lowell, Massachusetts 01852. E-mail: juzi_zhao@uml.edu.
- Y. Xiang is with the Department of Cloud Platform Software Research, AT&T Labs, Bedminster, NJ 07921. E-mail: yxiang@research.att.com.
- T. Lan, H. Huang and S. Subramaniam are with the Department of Electrical and Computer Engineering, The George Washington University, Washington, DC 20052. E-mail: {tlan, howie, suresh}@gwu.edu.

Manuscript received 31 Aug. 2015; revised 1 Mar. 2016; accepted 2 May 2016. Date of publication 20 May 2016; date of current version 18 Jan. 2017. Recommended for acceptance by D. Arnold.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2016.2571281

TABLE 1
Main Notation

Symbol	Meaning
n	Number of tasks, indexed by $i = 1, \ldots, n$
m_i	Number of VMs for task i
R_i	Reliability of task i
$U_i(R_i)$	Utility of task i as a function of R_i
$T_{v,i}, \eta_i$	Checkpoint interval and initial offset of task i
$T_{s,i}$	Checkpoint overhead of task i
$T_{n,i}$	The time to pause and save local VM images of task i
$T_{b,i}$	The time to transfer images to destinations of task i
T_r	VM rollback and recovery time
$\mathbf{X}_i$	Checkpoint routing vector of task i
$\mathbf{Y}_i$	VM placement vector of task i
$\mathcal{P}$	The set of feasible checkpoint routing vectors
$I_i(T_{v,i})$	VM delta image size as a function of $T_{v,i}$
$\mathbf{G}^b, \mathbf{G}^o$	Background traffic and I/O vector
$\mathbf{C}^b, \mathbf{C}^o$	Data center network and I/O capacity constraints
B_i, O_i	Network bandwidth and I/O assigned for task i
λ	Node failure rate
$\mathbf{V}(t)$	Lagrangian multiplier for network capacity constraints
$\mathbf{W}(t)$	Lagrangian multiplier for I/O capacity constraints

switches are highly congested [11], which leaves adequate bandwidth among local switches for peer-to-peer traffic. Our approach can effectively convert under-utilized network resources into an on-demand reliability service, which can be purchased by users on demand. The goal of our paper is to provide a novel framework for elastic reliability optimization through a commonly-used technique checkpointing, which offers a theoretical support for enabling Reliability as a Service for different cloud applications. A prototype and standard APIs would be interesting future work to explore in our next paper on the topic.

The rest of the paper is organized as follows: The system model is presented in Section 2. In Section 3, we formulate the joint reliability maximization problem as an optimization with heterogeneous reliability utilities. In Section 4, we investigate the combinatorial structure of this problem and make use of dual-decomposition [12], [44] to propose an efficient algorithm for the joint reliability maximization problem. In Section 5, we evaluate our algorithms. It is shown that our solution improves reliability by an order of magnitude over both random and centralized checkpointing mechanisms. Section 6 presents concluding remarks and directions for future work.

2 System Model

2.1 VM Checkpointing in Data Centers

Checkpointing is a typical fault tolerance technique in distributed systems and high-performance computing. In current data centers, Virtual Machine Monitors (VMMs) are capable of checkpointing the states of its VMs. VMMs can take local and uncoordinated checkpoints independently of each other. However, this runs the risk of cascaded rollbacks if causality is not respected. To avoid this, when a task comprises multiple VMs, synchronous VM checkpoints are taken, as shown in Fig. 1, so that they can be rolled back to the same point of execution. Checkpoint scheduling algorithms for optimizing reliability of *a single job* have been proposed in [7], [8], [9], [10]. To amortize high overhead,

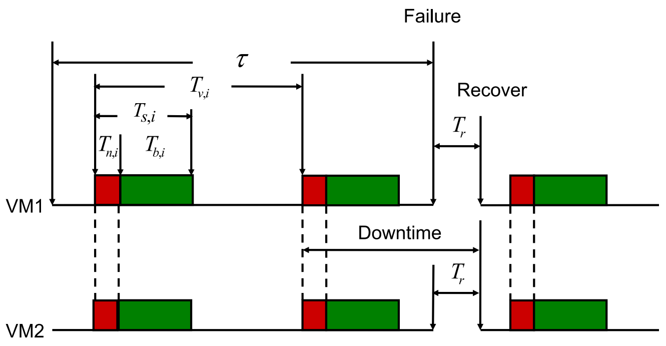


Fig. 1. VM checkpoint and recovery model for a single job.

checkpoint intervals are often chosen to be large as long as rollback costs are acceptable.

Yet these solutions fall short in optimizing checkpoints of multiple jobs that require different reliabilities, due to the lack of a model for checkpoint interference among jobs. In multi-job scenarios, uncoordinated job checkpoints taken independently of each other run the risk of interfering with each other if a joint checkpoint scheduling is not enabled. In this paper, we assume that VMMs support a coordinated checkpointing mechanism. For a task i with multiple VMs, checkpointing the task means synchronously checkpointing all its VMs. We treat the individual VM checkpoints as a single checkpoint event with overhead $T_{s,i} = T_{n,i} + T_{b,i}$, where $T_{n,i}$ is the time to pause and save local VM images and $T_{b,i}$ denotes the time to transfer the images to remote destinations as shown in Fig. 1. In this paper, we consider this general model that separates VM checkpointing and transferring, since $T_{n,i}$ and $T_{b,i}$ are determined by I/O and network bandwidth respectively; they can be combined into a single process in practice, e.g., when real-time VM migration technique is used.

Joint checkpoint scheduling of multiple jobs is critical for improving datacenter job reliability, since the more frequent the checkpoints, the less the downtime each job receives. It is easy to see that reliability is greatly affected by checkpoint interference and overhead: $T_{n,i}$, the time to save local checkpoint images, depends on how I/O resources are shared, and $T_{b,i}$, the time to transfer saved images, relies on how network resources are shared [29], [30]. Upon a failure at time τ during the kth interval, a recovery T_r is performed to restart a job form its latest available checkpoints with timestamp $(k-1)T_{v,i}$. We consider a time-slotted model, so that a system snapshot is taken every Δt seconds.

In the theoretical analysis, our assumptions include: (1) each checkpoint involves overhead $T_{s,i} = T_{n,i} + T_{b,i}$, where $T_{n,i}$ is the time to pause and save local VM images and $T_{b,i}$ denotes the time to transfer the images to remote destinations; (2) we consider a time-slotted model, and (3) failures of hosts (nodes) are modeled by a Poisson process with known rate λ. Assumption (1) is a general model that separates VM checkpointing and transferring, since $T_{n,i}$ and $T_{b,i}$ are determined by I/O and network bandwidth respectively, while they can be combined into a single process in practice, e.g., when real-time VM migration technique is used. Similar models are used in [7], [8], [31]. Assumption (2) makes the optimization tractable and is routinely used in existing work on scheduling and optimization [32], [33]. Assumption (3) is standard for reliability analysis, such as [34], [35], [36], [37].

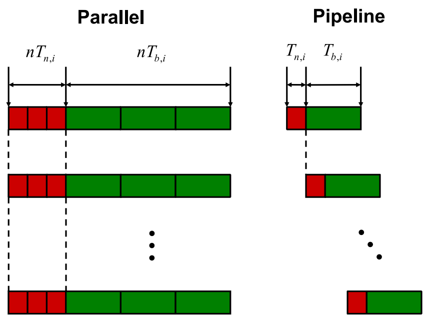


Fig. 2. Coordinated checkpointing of multiple jobs mitigates overhead and improves reliability.

As shown in Fig. 2, without joint scheduling, the jobs may take conflicting, parallel checkpoints and evenly share a bottleneck. The overhead for saving and transferring checkpoint images is significantly amplified due to interference. Alternatively, performing a joint checkpoint scheduling, such as the pipeline scheme shown in Fig. 2, mitigates interference and optimizes reliability. A preliminary experiment with four jobs in Section 5 will show that pipeline scheduling reduces downtime by as much as one order of magnitude over parallel checkpoints.

2.2 Quantifying Reliability

To achieve high reliability, VM checkpoint images must be placed in different clusters and/or availability zones, whose failures are made as independent as possible. Our research is strongly motivated by recent major service outages, some of which are recapped below.

- In February 2010, Google experienced a power failure that affected 25 percent of the machines in one of its datacenters [13].
- In March 2011, Amazon's cloud-hosted Web Services experienced a catastrophic failure due to configuration error, knocking down hundreds of sites off the web [14].
- In February 2012, a process meant to detect failed hardware in Microsoft's Azure cloud was inadvertently triggered by a software bug and caused 7.5 hours downtime in 4 regional datacenters [15].
- In March 2012, a network card failure affected 2 core routers in OVH cloud and the backbone network was down for 2 hours.

To utilize the diversity of datacenter topologies and availability zones, reliability optimization needs to determine not only how often checkpoints are created, but also where they are placed. We consider the placement of checkpoints into different clusters and availability zones, which are assumed to have independent and identical failure probabilities. After each host failure, tasks can be recovered from the latest available checkpoints. All tasks using the failed node must be rolled back and restarted. We assume that failures of hosts (nodes) are modeled by a Poisson process with known rate λ. Therefore, the mean time between failures is $1/\lambda$. In this research, we consider infinite time-horizon jobs whose reliability given by the expected fraction of service downtime, where the expectation is taken over failure modes and distributions. As large-scale datacenters are typically well-managed and tracked for any critical

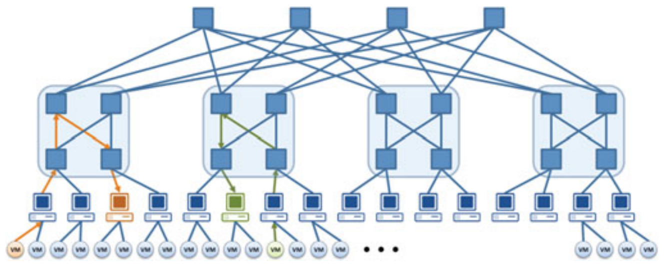


Fig. 3. Illustrations of peer-to-peer checkpointing with fat-tree topology, where traffics are distributed over the entire network.

events, the event logs can be used to provide important historical information for estimating the failure rates.

We quantify reliability as a function of failure rate λ, and checkpoint parameters, including checkpoint overhead $T_{s,i} = T_{n,i} + T_{b,i}$, checkpoint interval $T_{v,i}$, and rollback time T_r. For example, total downtime per failure in Fig. 1 is given by $kT_{n,i} + \tau - (k-1)T_{v,i} + T_r$. We define reliability by one minus the fraction of service downtime. This yields the following Lemma 1 on the expected reliability with periodic checkpointing.

Lemma 1. *If VMs of task i reside on h_i different hosts, the expected reliability of task i with periodic checkpointing interval $T_{v,i}$ is*

$$\begin{aligned} R_i = 1 &- \sum_{k=1}^{\infty} \int_0^{T_{s,i}} \frac{t + kT_{n,i} + T_r + T_{v,i}}{kT_{v,i}} f_k(t)dt \\ &- \sum_{k=1}^{\infty} \int_{T_{s,i}}^{T_{v,i}} \frac{t + kT_{n,i} + T_r}{kT_{v,i}} f_k(t)dt, \end{aligned} \tag{1}$$

where $f_k(t) = h_i \lambda e^{-h_i\lambda[t+(k-1)T_{v,i}]}$ is the probability that a VM failure for task i occurs t seconds after the kth checkpoint interval.

Proof. Since task i uses h_i hosts, its VM failure process is Poisson with rate $h_i\lambda$. Therefore, $f_k(t) = h_i \lambda e^{-h_i\lambda[t+(k-1)T_{v,i}]}$ is the p.d.f of VM failure at time $t + (k-1)T_{v,i}$.

Now if the failure occurs during $[T_{s,i}, T_{v,i}]$ of the kth checkpoint interval, the total service downtime in $kT_{v,i}$ seconds is $t + kT_{n,i} + T_r$, where the checkpointing overhead $T_{n,i}$ is experienced in all checkpoint intervals due to pausing all VMs. In contrast, if the failure occurs during $[0, T_{s,i}]$ of the kth checkpoint interval, the total service downtime becomes $t + kT_{n,i} + T_r + T_{v,i}$, because the kth checkpoint has not been completed yet, and task i must roll-back to the $(k-1)$th checkpoint. Therefore, reliability is obtained as the mean fraction of service uptime as in (1). This completes the proof of Lemma 1. □

2.3 Peer-to-Peer Checkpointing to Mitigate Congestion

Mechanisms for checkpointing on central storage servers have been provided in [4], [5], [6]. Since a huge amount of VM image data must be transferred periodically, as the number of tasks and VMs increase in a data center, the links that connect central storage server and core switches easily become congested. To avoid such a bottleneck, we propose a peer-to-peer checkpointing mechanism that enables VM images to be transferred and saved among neighboring peers. Fig. 3 shows a schematic diagram of peer-to-peer checkpointing. In comparison, in a centralized checkpointing scheme where networked storage servers are connected to top-level switches, all checkpointing traffic is routed through core switches.

To characterize the benefits of peer-to-peer checkpointing, we first notice that if we further assume that checkpoint interval $T_{v,i}$ is much smaller than the mean time between failures, i.e., $T_{v,i} \ll 1/(h_i\lambda)$, then reliability can be approximated by the following lemma:

Lemma 2. *When $T_{v,i} \ll 1/(h_i\lambda)$, reliability R_i can be approximated by*

$$R_i = 1 - \frac{T_{n,i}}{T_{v,i}} - h_i\lambda\left(\frac{T_{v,i}}{2} + T_r + T_{s,i}\right). \tag{2}$$

Proof. This result is straightforward by applying the approximation $e^{-h_i\lambda t} = 1$ to $f_k(t)$ on the right hand side of (1), since $t \le T_{v,i} \ll 1/(h_i\lambda)$. □

To illustrate limitations of the centralized checkpointing method, we consider a scenario where the link connecting central storage servers and top-level core switches is the only traffic bottleneck. This analysis provides an upper bound for the centralized checkpointing method because possible local bottlenecks are ignored. Suppose that there are n tasks with the same checkpoint interval $T_{v,i}$ and VM image size I_i. The aggregate checkpoint traffic from all tasks cannot exceed the total capacity C over a checkpoint interval, i.e.,

$$\sum_{i=1}^{n} m_i I_i \le C T_{v,i}. \tag{3}$$

According to (2), it implies that, for centralized checkpointing,

$$R_i \le 1 - h_i\lambda\frac{T_{v,i}}{2} \le 1 - \frac{h_i\lambda I_i}{2C}\sum_{i=1}^{n} m_i. \tag{4}$$

Reliability R_i tends to zero as the number of VMs $\sum_{i=1}^{n} m_i$ grows large. The centralized checkpointing method leads to very poor performance for large-scale data centers, where a finite bandwidth from central storage servers is shared by a large number of VM checkpoints. This does not pose a problem for peer-to-peer checkpointing, because checkpoint traffics may be distributed over local links at low-level switches, which also scale up when data center size increases. Therefore, this approach is much more.

3 Joint Reliability Optimization

3.1 Problem Formulation

In this paper, we focus on how to determine optimal checkpoint scheduling and routing under I/O and network capacity constraints. In our model, checkpoint scheduling is decided by periodic checkpoint intervals $T_{v,i}$ and initial time offset η_i, while checkpoint routing is determined by the selection of checkpoint destination nodes and traffic routing among peers, collectively denoted by $\mathcal{P}$.

We use a utility function $U_i(\cdot)$ to model the reliability requirement of task i. A survey [2] showed that 45 percent of cloud users are satisfied with a 99.9 percent reliability guarantee (i.e., 45 minutes unplanned downtime per month), while 14 percent would pay at least 25 percent more to get a 99.99 percent reliability guarantee (i.e., approximately 4 minutes unplanned downtime per month), and only 6 percent would pay at least 50 percent more to go beyond 99.99 percent. Therefore, $U_i(R_i)$ is assumed to be an increasing function of R_i. For instance, we can choose $U_i(R_i) = -w_i \log_{10}(1 - R_i)$, where w_i are user-specific weights.

In order to create a checkpoint, only a *delta disk* [7] that contains incremental VM changes after the last checkpoint has to be saved and transferred, once the first checkpoint is done. This process considerably reduces the time needed to make the checkpoint. Therefore, we consider variable VM image sizes, as a non-decreasing function of checkpoint interval, e.g., a logarithm function $I_i(T_{v,i}) = a\log(T_{v,i}) + b$ where a, b are appropriate constants. The time to take a checkpoint and transfer its images, $T_{n,i}$ and $T_{b,i}$, can be computed by *delta disk* size $I_i(T_{v,i})$ and allocated I/O O_i, bandwidth B_i:

$$T_{n,i} = \left\lceil \frac{I_i(T_{v,i})}{O_i\Delta t} \right\rceil \text{ and } T_{b,i} = \left\lceil \frac{I_i(T_{v,i})}{B_i\Delta t} \right\rceil. \tag{5}$$

For VM checkpointing, it is easy to see that task i generates periodic I/O usage for all

$$t \in [\eta_i + kT_{v,i}, \eta_i + kT_{v,i} + T_{n,i}],\ \forall k \in Z_+, \tag{6}$$

as well as periodic network traffic for all

$$t \in [\eta_i + kT_{v,i} + T_{n,i}, \eta_i + kT_{v,i} + T_{s,i}],\ \forall k \in Z_+. \tag{7}$$

Therefore, increasing checkpoint frequency (i.e., reducing $T_{v,i}$) increases checkpoint traffic proportionally.

Network Constraints. Consider a data center with L links, indexed by $l = 1, \ldots, L$, each with a fixed capacity C_l^b. We define a checkpoint routing vector $\mathbf{X}_i$ of length L for task i by

$$\mathbf{X}_{i,l} = \begin{cases} x, & \text{if } x \text{ VM images of task } i \text{ transverse link } l, \\ 0, & \text{otherwise.} \end{cases}$$

We assume that this checkpoint routing vector $\mathbf{X}_i$ remains unchanged for the entire duration of task i. Let B_i be the checkpoint network bandwidth assigned to each VM of task i. Combining (Eq. (7)) and the definition of checkpoint routing vector $\mathbf{X}_i$, we can formulate a network capacity constraint as follows:

$$\mathbf{G}^b + \sum_{i=1}^{n} B_i \mathbf{X}_i \mathbf{1}_i^b(t) \le \mathbf{C}^b,\ \forall t, \tag{8}$$

where $\mathbf{C}^b = [C_1, \ldots, C_L]$ is a set of link capacity constraints, and $\mathbf{1}_i^b(t)$ is an indicator function defined by

$$\mathbf{1}_i^b(t) = \mathbf{1}_{\{t \in [\eta_i + kT_{v,i} + T_{n,i}, \eta_i + kT_{v,i} + T_{s,i}], \forall k\}}. \tag{9}$$

Here $\mathbf{G}^b = [G_1^b, \ldots, G_L^b]$ is a background traffic vector, representing the link capacities set aside for normal task traffic. An empirical measurement study in [11] shows that average traffic per VM is stable at large time-scales. Thus, we treat $\mathbf{G}^b$ as a time-invariant vector, where G_l^b denotes the aggregate task traffic on link l.

I/O Constraints. Similarly, consider J hosts, indexed by $j = 1, \ldots, J$, each with a fixed I/O capacity C_j^o. We define a VM placement vector $\mathbf{Y}_i$ of length J for task i by

$$\mathbf{Y}_{i,j} = \begin{cases} y, & \text{if } y \text{ VMs of task } i \text{ are placed on host } j, \\ 0, & \text{otherwise.} \end{cases}$$

Let O_i be the checkpoint I/O assigned to each VM of task i. Combining (6) and $\mathbf{Y}_i$, we can formulate an I/O capacity constraint as follows:

$$\mathbf{G}^o + \sum_{i=1}^{n} O_i \mathbf{Y}_i \mathbf{1}_i^o(t) \leq \mathbf{C}^o, \ \forall t, \tag{10}$$

where $\mathbf{1}_i^o(t)$ is an indicator function defined by the time interval for saving local checkpoints:

$$\mathbf{1}_i^o(t) = \mathbf{1}_{\{t \in [\eta_i + kT_{v,i}, \eta_i + kT_{v,i} + T_{n,i}], \forall k\}}. \tag{11}$$

Here $\mathbf{G}^o = [G_1^o, \ldots, G_J^o]$ is a background I/O vector, representing the I/O capacities set aside for normal task traffic.

Combining (2), (5), (8), (9) (10), and (11), we then formulate the Joint Checkpoint Scheduling and Routing (JCSR) problem under network and I/O capacity constraints:

$$\text{maximize} \tag{12}$$

$$\sum_{i=1}^{n} U_i(R_i), \tag{13}$$

$$\text{subject to} \tag{14}$$

$$R_i = 1 - \frac{T_{n,i}}{T_{v,i}} - h_i \lambda \left(\frac{T_{v,i}}{2} + T_r + T_{s,i} \right), \tag{15}$$

$$\mathbf{G}^b + \sum_{i=1}^{n} B_i \mathbf{X}_i \mathbf{1}_i^b(t) \leq \mathbf{C}^b, \ \forall t, \tag{16}$$

$$\mathbf{G}^o + \sum_{i=1}^{n} O_i \mathbf{Y}_i \mathbf{1}_i^o(t) \leq \mathbf{C}^o, \ \forall t, \tag{17}$$

$$T_{s,i} = T_{n,i} + T_{b,i}, \tag{18}$$

$$\text{variables} \tag{19}$$

$$\eta_i, T_{v,i} \in \mathcal{T}, B_i, O_i, \mathbf{X}_i \in \mathcal{P}. \tag{20}$$

Here we only allow users to choose $T_{v,i}$ from a finite set of checkpoint intervals, $\mathcal{T} = \{T_1, T_2, \ldots, T_z\}$. Similarly, we use $\mathcal{P}$ to denote the set of all feasible checkpoint routing vectors. Constraints (16) and (17) ensures that the required checkpoint and task traffic can be supported. $T_{n,i}$ and $T_{b,i}$ are determined by VM image sizes and I/O, network bandwidth assignments O_i, B_i, respectively. The VM placement $\mathbf{Y}_i$ is an input parameter. Our optimization problem formulation uses a general system model, where any nodes and switches (both top-of-rank and aggregate switches) can be used for VM checkpointing. In particular, $\mathbf{X}_i$ and $\mathbf{Y}_i$ are the checkpoint routing vector and the VM placement vector of task i. Whether a link, a switch or a node is chosen for checkpointing depends on the optimal solution of our proposed optimization.

3.2 Finding Interference-Free Schedule is NP Hard

In this section, we prove that the JCSR problem is NP-hard in general and present a few special cases in which the problem can be solved optimally.

Lemma 3. *The JCSR problem is NP-hard.*

Proof. We show that if the JCSR problem can be solved in polynomial time, then the f-edge coloring problem for multigraphs can also be solved in polynomial time, which results in a contradiction to the NP-hardness of f-edge coloring problem. Toward this end, we consider a special case of the JCSR problem with the following simplifications:

- the tasks are identical, each with a single VM (i.e., $h_i = 1$)
- checkpoint interval (T_v) is the same for all tasks
- the link bandwidth (B) allocated to each task is the same, i.e., $T_{b,i} = T_b \ \forall i$
- the I/O (O) allocated to each task is the same, i.e., $T_{n,i} = T_n \ \forall i$
- there exists sufficient bandwidth and I/O capacity except for the links directly connecting hosts and switches
- the checkpoint destination for each task is fixed and known.

In this special case, all tasks are homogeneous, and should receive the same checkpoint interval T_v. For each value of T_v, we can calculate T_n and T_b. Together with T_r, $h = 1$, and λ, we can calculate the reliability R and utility $U(R)$. Thus, the JCSR problem is reduced to a verification problem: whether we can schedule the checkpoints for all tasks within a particular T_v (by choosing a proper offset η_i for each task i).

According to our assumption that there exists sufficient bandwidth and I/O capacity except for the links directly connecting hosts and local switches, the only constraints in the JCSR problem are the link bandwidth available between hosts and local switches. Based on the bandwidth (B) assigned to each task, and the available bandwidth of each link, there is an upper bound on the number of checkpoints images that can be sent/received by each host j at each time slot, which we denote as z_j. We argue that problem JCSR is in NP, since given a collection of η_i, it can be efficiently checked that whether the η_i values are feasible according to z_j of each host j and T_v (given T_b and T_n). Now it only remains to prove that if the JCSR problem in this special case can be solved in polynomial time, so is the f-edge coloring problem for multigraphs.

We consider a given instance of f-edge coloring problem and converts it into an equivalent JCSR problem. Let $G(V, E)$ be a graph, f be an integer function on V with $f(v)$ defined for each vertex $v \in V$. An f-edge coloring problem asks whether it is possible to color the edges of G using at most k different colors such that each color appears at vertex v at most $f(v)$ times. It is well known as an NP-hard problem [16], [17]. Given an f-edge coloring problem, we convert it into an instance of the JCSR problem as follows: Each vertex v in G is represented by a host j in the data center. If there is an edge between two vertexes v_1 and v_2 in G, then there exists a VM (task) hosted by j_1 whose checkpointing destination is host j_2. Next, for a vertex v with value $f(v)$, we set the bandwidth between host j and its edge switch as $f(v)B$ (i. e., $z_j = f(j)$). This construction can be done in polynomial time based on the size of the given f-edge coloring problem.

Finally, it is easy to show that if we can solve the JCSR problem with $T_v \leq T_b k + T_n$, then we can also solve the f-edge coloring problem with k colors. Suppose that a feasible solution to the JCSR problem is found. For all the tasks have same η, we assign the corresponding edges with the same color. Thus, each color appears at vertex v at most $f(v) = z_j$ times. Since each task takes T_b time slots to transfer checkpoint image, $T_v \leq T_b k + T_n$ means that for each host, there are at most k tasks with different η values (with that host as the tasks' VM placement or as their checkpoint destination). Therefore, the corresponding vertex v is assigned at most k different colors. Since the f-edge coloring problem is NP-hard, we conclude that the JCSR problem is also NP-hard. □

While a general solution to the JCSR problem cannot be obtained in polynomial time, we present a few special cases where the JCSR problem can be solved optimally.

Remark 1. Consider the special case introduced in the proof of Lemma 3, where all tasks are homogeneous and have the same checkpoint interval, I/O and bandwidth allocation. Each task has only a single VM and its checkpoint destination is fixed. There are sufficient bandwidth and I/O capacity except for the links directly connecting hosts and local switches. In addition, we assume that all hosts in the data center are partitioned into two disjoint sets, one consisting of (computing) hosts only serving VMs, the other consisting of (storage) hosts only used as checkpoint destinations. In this case, the problem can be solved as f-edge coloring for bipartite graphs, which can be optimally solved in polynomial time using algorithms such as [16].

Remark 2. Again, consider the special case introduced in the proof of Lemma 3. In addition, we assume that the bandwidth for checkpointing on the links are divided into two parts - one dedicated to transmitting checkpoint image to checkpoint destination and the other dedicated to receiving checkpoint image from VM placement (i.e., out-going and incoming bandwidth). Then, the JCSR problem can also be optimally solved as f-edge coloring for bipartite graphs.

Remark 3. Consider a special case with only I/O constraints and only one VM per task. The offset η_i is assumed to be zero for all tasks. Then the VM placements are independent with each other. Since link bandwidth are sufficiently large, we have $T_{b,i}$=1 slot for each task i. For each node j, we only need to decide the O_i^j allocated to each task i at node j to have the maximum aggregate utility. The problem can be solved under three separate conditions: (i) If all the tasks have same and fixed checkpointing interval T_v (in turn, the same checkpoint image size I), then the maximum summation of utility U is a function of the O_i^j (Eq. (15)), as $\sum_i I/O_i^j$ decreases, U increases, and $\sum_i O_i^j \leq C^o$. The optimal value is obtained when C^o are equally allocated to all tasks i. (ii) If the sizes of checkpointing images for the tasks are different, then U is a function of $\sum_i I_i/O_i^j$. Let $Q_i = I_i/O_i^j$ (i.e., $O_i^j = I_i/Q_i$), so we want to minimize $\sum_i Q_i$ with constraint $\sum_i I_i/Q_i \leq C^o$. The optimal value is obtained when all the Q_i are same for all tasks i, i.e., $Q_i = C^o/\sum_i I_i$. (c) If each task has multiple VMs x_i, and they are placed on the same node, then this case can be derived easily by assigning each task a new $I_i' = I_i x_i$, then use I_i' instead of I_i in cases (a and b).

4 Solving the Joint Checkpoint Scheduling and Routing Problem

4.1 Our Solution Using Dual Decomposition

The problem (13) is a non-convex and combinatorial optimization and there is no computationally-efficient solution even in a centralized manner. In this paper, we leverage the technique of dual-decomposition in [12], [44] to obtain a sub-optimal solution. Among many choices of heuristic methods, the one we develop below has the advantage of allowing a distributed implementation without cooperation of different tasks.

Let M be the least common multiple of all feasible checkpoint intervals in $\mathcal{T} = \{T_1, T_2, \ldots, T_z\}$. Due to our model of periodic checkpointing, it is sufficient to consider the network capacity constraint in (16) over $[0, M]$. Let $\mathbf{V}(t)$ and $\mathbf{W}(t)$ be Lagrangian multipliers vector for the network and I/O capacity constraints, which are both time-dependent. We derive the Lagrangian for the joint checkpoint scheduling and routing problem in Eq. (21) on next page.

$$\begin{aligned}\mathcal{L} = \sum_{i=1}^{n} U_i(R_i) - \sum_{t=0}^{M} \mathbf{V}(t)^T \left[\mathbf{G}^b + \sum_{i=1}^{n} B_i \mathbf{X}_i \mathbf{1}_i^b(t) - \mathbf{C}^b \right] \\ - \sum_{t=0}^{M} \mathbf{W}(t)^T \left[\mathbf{G}^o + \sum_{i=1}^{n} O_i \mathbf{Y}_i \mathbf{1}_i^o(t) - \mathbf{C}^o \right].\end{aligned} \tag{21}$$

The other two constraints (15) and (18) can be easily substituted in the Lagrangian above and are suppressed for a simple presentation.

Since M is an integer multiple of $T_{v,i}$, we have

$$\begin{aligned}&\sum_{t=0}^{M} \mathbf{V}(t)^T \left[\sum_{i=1}^{n} B_i \mathbf{X}_i \mathbf{1}_i^b(t) \right] \\ &= \sum_{i=1}^{n} B_i \sum_{t=0}^{M} \mathbf{V}(t)^T \mathbf{X}_i \mathbf{1}_i^b(t) \\ &= \sum_{i=1}^{n} \frac{I_i(T_{v,i})}{T_{b,i}\Delta t} \sum_{t=0}^{M} \mathbf{V}(t)^T \mathbf{X}_i \mathbf{1}_i^b(t),\end{aligned} \tag{22}$$

where the last step uses $I_i(T_{v,i}) = B_i T_{b,i} \Delta t$ in (5). Similarly, we have

$$\begin{aligned}&\sum_{t=0}^{M} \mathbf{W}(t)^T \left[\sum_{i} O_i \mathbf{Y}_i \mathbf{1}_i^o(t) \right] \\ &= \sum_{i=1}^{n} O_i \sum_{t=0}^{M} \mathbf{W}(t)^T \mathbf{Y}_i \mathbf{1}_i^o(t) \\ &= \sum_{i=1}^{n} \frac{I_i(T_{v,i})}{T_{n,i}\Delta t} \sum_{t=0}^{M} \mathbf{W}(t)^T \mathbf{Y}_i \mathbf{1}_i^o(t),\end{aligned} \tag{23}$$

where the last step uses $I_i(T_{v,i}) = O_i T_{n,i} \Delta t$ in (5).

Plugging (22) and (23) into the Lagrangian Eq. (21), we obtain

$$\mathcal{L} = \sum_{i=1}^{n}\left[U_i(R_i) - I_i(T_{v,i})\left(\frac{1}{T_{b,i}\Delta t}\bar{\mathbf{V}}_i^T\mathbf{X}_i + \frac{1}{T_{n,i}\Delta t}\bar{\mathbf{W}}_i^T\mathbf{Y}_i\right)\right] + \sum_{t=0}^{M}\mathbf{V}(t)^T\left[\mathbf{C}^b - \mathbf{G}^b\right] + \sum_{t=0}^{M}\mathbf{W}(t)^T[\mathbf{C}^o - \mathbf{G}^o],$$

where

$$\bar{\mathbf{V}}_i^T = \sum_{t=0}^{M}\mathbf{V}(t)^T\mathbf{1}_i^b(t)$$
$$\bar{\mathbf{W}}_i^T = \sum_{t=0}^{M}\mathbf{W}(t)^T\mathbf{1}_i^o(t). \tag{24}$$

Now, for given Lagrangian multipliers $\mathbf{V}(t)$ and $\mathbf{W}(t)$, the optimization of $\mathcal{L}$ over checkpoint scheduling and routing is decoupled into n individual sub-problems:

$$\max_{\eta_i, T_{v,i}, B_i, O_i, \mathbf{X}_i} U_i(R_i) - I_i(T_{v,i}) \left(\frac{1}{T_{b,i}\Delta t}\bar{\mathbf{V}}_i^T\mathbf{X}_i + \frac{1}{T_{n,i}\Delta t}\bar{\mathbf{W}}_i^T\mathbf{Y}_i\right), \ \forall i. \tag{25}$$

Here, the checkpoint sequence offset η_i only affects average congestion prices $\bar{\mathbf{V}}_i^T$ and $\bar{\mathbf{W}}_i^T$, while B_i, O_i and $\mathbf{X}_i$ are determined by checkpoint routing/placement decisions. Thus, to solve (25) sub-optimally, we can iteratively optimize it over two sets of variables: $\{B_i, O_i, \mathbf{X}_i\}$, and $\{\eta_i, T_{v,i}\}$, respectively. This results in the design of a heuristic and distributed algorithm for solving problem (13), if the Lagrangian multipliers $\mathbf{V}(t)$ and $\mathbf{W}(t)$ are updated by a gradient method:

$$\mathbf{V}^{g+1}(t) = \left[\mathbf{V}^g(t) + \mu_g\left(\mathbf{G}^b + \sum_{i=1}^{n}B_i\mathbf{X}_i\mathbf{1}_i^b(t) - \mathbf{C}^b\right)\right]^+, \forall t, \tag{26}$$

$$\mathbf{W}^{g+1}(t) = \left[\mathbf{W}^g(t) + \mu_g\left(\mathbf{G}^o + \sum_{i=1}^{n}O_i\mathbf{Y}_i\mathbf{1}_i^o(t) - \mathbf{C}^o\right)\right]^+, \forall t, \tag{27}$$

where g is the iteration number and μ_g is a proper step-size. The initial values of V and W are set as

$$\mathbf{V}^0(t) = \left[\frac{\mathbf{C}^b}{\mathbf{C}^b - \mathbf{G}^b}\right]^+, \forall t, \tag{28}$$

$$\mathbf{W}^0(t) = \left[\frac{\mathbf{C}^o}{\mathbf{C}^o - \mathbf{G}^o}\right]^+, \forall t. \tag{29}$$

4.2 Algorithm Solution for Reliability Optimization

We next present a heuristic algorithm that finds a sub-optimal solution for the joint checkpoint scheduling and routing problem, leveraging the dual decomposition method presented above. The key idea is to iteratively compute the individual-user optimization problem in (25) and the price vector update in (26) and (27). To reduce search complexity, we further break down the individual-user optimization problem in (25) into two sub-problems, over $\{B_i, O_i, \mathbf{X}_i\}$, and $\{\eta_i, T_{v,i}\}$, respectively. The Dijkstra algorithm is used to find the optimal routing vector $\mathbf{X}_i$ with link cost $\bar{\mathbf{V}}_i$ and host cost $\bar{\mathbf{W}}_i$. For a chosen tolerance ϵ, the proposed algorithm is summarized in Algorithm 1.

Algorithm 1. Joint Checkpoint Scheduling and Routing to Maximize Reliability

```
Step 1:
1. Initialize random interval $T_{v,i}$ and offset $\eta_i$
2. Initialize random routing vector $\mathbf{X}_i$ and feasible bandwidth $B_i, O_i$
3. Initialize $V$ and $W$
$V = \left[\frac{\mathbf{C}^b}{\mathbf{C}^b-\mathbf{G}^b}\right]^+$, $W = \left[\frac{\mathbf{C}^o}{\mathbf{C}^o-\mathbf{G}^o}\right]^+$
Step 2: Solve individual-user optimization problem in (25):
for $0 \le i \le n$ do
  Step 2.a: Solve optimal $B_i$, $O_i$ and $\mathbf{X}_i$:
  Calculate $\bar{\mathbf{V}}_i$ and $\bar{\mathbf{W}}_i$ according to (24)
  Treat $\bar{\mathbf{V}}_i$ as link costs, $\bar{\mathbf{W}}_i$ as host costs
  $\mathbf{X}_i \leftarrow Dijkstra$ for all VMs
  for $T_{n,i} \in [1, T_{v,i} - 1]$ do
    for $T_{b,i} \in [1, T_{v,i} - T_{n,i}]$ do
      $O_i = \frac{I_i(T_{v,i})}{T_{n,i}\Delta t}$ and $B_i = \frac{I_i(T_{v,i})}{T_{b,i}\Delta t}$
      check if $O_i$ and $B_i$ resources are available according to $\mathbf{Y}_i$, $\mathbf{X}_i$, $\eta_i$, $T_{n,i}$ and $T_{b,i}$
      find $T_{n,i,opt}$ and $T_{b,i,opt}$ to maximize
      $U_i(R_i) - I_i(T_{v,i})\left(\frac{1}{T_{b,i}\Delta t}\bar{\mathbf{V}}_i^T\mathbf{X}_i + \frac{1}{T_{n,i}\Delta t}\bar{\mathbf{W}}_i^T\mathbf{Y}_i\right)$
    end for
  end for
  $T_{n,i} \leftarrow T_{n,i,opt}$, $T_{b,i} \leftarrow T_{b,i,opt}$
  Step 2.b: Search for optimal $T_{v,i}$ and $\eta_i$:
  Calculate $\bar{\mathbf{V}}_i$ and $\bar{\mathbf{W}}_i$ according to (24).
  for $T_{v,i} \in \mathcal{T}$ do
    for $\eta_i \in [0, T_{v,i}]$ do
      Find $T_{v,i,opt}$ and $\eta_{i,opt}$ to maximize
      $U_i(R_i) - I_i(T_{v,i})\left(\frac{1}{T_{b,i}\Delta t}\bar{\mathbf{V}}_i^T\mathbf{X}_i + \frac{1}{T_{n,i}\Delta t}\bar{\mathbf{W}}_i^T\mathbf{Y}_i\right)$
    end for
  end for
  $T_{v,i} \leftarrow T_{v,i,opt}$, $\eta_i \leftarrow \eta_{i,opt}$
end for
Step 3: Update price vector $\mathbf{V}(t)$ and $\mathbf{W}(t)$:
1. $\mathbf{V}(t) \leftarrow \mathbf{V}^{g+1}(t)$ according to (26).
2. $\mathbf{W}(t) \leftarrow \mathbf{W}^{g+1}(t)$ according to (27).
Step 4:
1. Record current reliability $R_i^0 \leftarrow R_i$
2. Compute new $R_i$ according to (15)
if $\sum_i |R_i - R_i^0| > \epsilon$ then
  Goto Step 2.
end if
```

Let us denote by N the number of nodes (including hosts and switches) in the topology; E the number of unidirectional links in the network; H the number of hosts in the network; n the number of tasks; T' the longest checkpoint interval (in time slots); M' the least common multiple of checkpoint intervals (in time slots); W the maximum number of VMs of each task; G the number of checkpointing interval choices; and I the number of iterations before the algorithm terminates. The time complexity of Algorithm 1 is analyzed as follows. Step 1 is called only once, while Steps 2-4 are called per iteration. The time complexities of

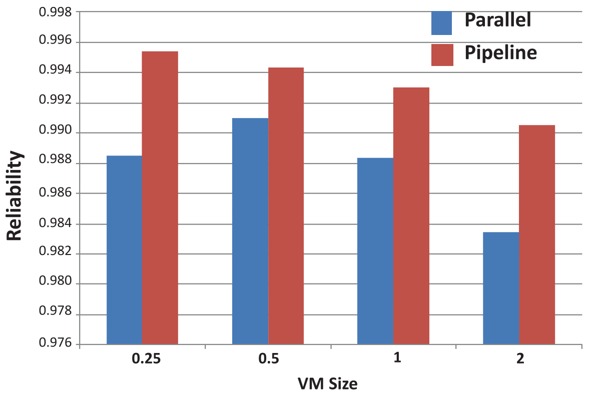


Fig. 4. Pipeline scheduling outperforms parallel scheduling when all tasks require the same reliability.

Steps 1-4 are $O(nT'^2(M'E + WN\log N + WE))$, $O(n(WN \log N + WE + T'^2M'E + GT'E))$, $O(nM'E)$, and $O(n)$ respectively, which make the total time complexity of the Algorithm 1 as $O(n(T'^2WN\log N + T'^2WE + IWN\log N + IWE + IT'^2M'E + IGT'E))$, which is linear in the number of tasks n.

5 Numerical Examples

5.1 Prototype Implementations

To demonstrate potential benefits of coordinated checkpoints, we constructed a 11-node local testbed with a single node serving 4 tasks, and conducted some preliminary tests on pipeline and parallel checkpoint scheduling. In this test, assuming that all tasks require equal reliability, pipeline scheduling shown in Fig. 4 allows checkpoints to be taken one after another and the image transfer time may be overlapped to transmit at the same time. In this case, each task can take full advantage of available I/O and bandwidth resources and achieve much higher reliability. On the other hand, in optimized parallel scheduling, the tasks may decide to take checkpoints at the same time, which would introduce performance interference as VM checkpointing consumes shared physical resources such as CPU and I/O. This in turn may also increase service downtime and result in lower reliability. As shown in Fig. 4, pipeline scheduling achieves significant reliability improvement as task VM-memory sizes vary from 0.25 GB to 2 GB.

5.2 Simulation Setup

In remaining of this section, we evaluate our design for joint checkpoint scheduling and routing on a Fat-tree topology [18], which consists of a collection of edge and aggregation switches that form a complete bipartite graph (see Fig. 3 as an example of fat-tree topologies). While data center traffic traces are generally proprietary and unavailable, recent studies [19], [20], [21] provide us a good characterization of traffic patterns inside data centers. The major objective of our evaluation is to move a step further than analysis and obtain empirically-validated insights about the feasibility/efficiency of providing reliability as a service in practical settings.

We construct a 1024-node Fat-tree topology. The nodes are connected to high speed switches (similar to Cisco Catalyst 4948 Switch), which offer a link capacity of $C_l = 10$Gbps for $l = 1, , L$ [43]. Each node represents a quad-core machine and can host up to 4 VMs. The total write I/O capacity of each host is about 80 MB/s. The part set aside for checkpointing is 25 or 50 percent. Δt is set as 5 seconds.

To incorporate task heterogeneity, we define two types of tasks: elephant tasks that comprise $m_i = 20$ VMs and generate large peer-wise flows uniformly distributed in $[50, 150]$Mbps, and mice tasks that comprise $m_i = 5$ VMs and generate small peer-wise flows uniformly distributed in $[0, 50]$Mbps. We randomly generate $n = 300$ tasks, each being an elephant task with probability 20 percent and a mice task with probability 80 percent. Background traffic vector $\mathbf{G}$ is constructed by randomly placing all VMs in the data center and employing a shortest-path algorithm to determine their traffic routing.

Each task is associated with a utility function, given by

$$U_i(R_i) = -w_i \log_{10}(1 - R_i), \tag{30}$$

where w_i is a user-specific weight uniformly distributed in $[0, 1]$. A larger weight implies a higher reliability demand and budget. We model checkpoint image size $I_i(T_{v,i})$ as increasing and convex functions of checkpoint interval $T_{v,i}$, i.e., $I_i(T_{v,i}) = (143 \cdot \log_{10} T_{v,i} - 254)$MB, the VM-memory sizes are obtained/tuned through measurements from a real implementation of checkpointing policies shown in Fig. 4. Further, we assume an average annual failure rate of one per node, which happens in most computing data centers [38], [39], a rollback time $T_r = 20$ seconds [40], and checkpoint interval $T_{v,i}$ is selected from $\mathcal{T} = \{20,000, 30,000, 40,000, 50,000\}$ seconds, so that checkpointing would be frequent enough to keep reasonable reliability level, and also not be too frequent to bring in longer downtime; numbers in this range are also validated to be appropriate given the failure rates in real cloud environments [41], [42].

5.3 Simulation Results

To provide benchmarks for our evaluations, we consider two heuristic algorithms with random selection of checkpoint intervals and checkpoint destinations. For each task i, offset η_i is uniformly distributed in $[0, T_{v,i}]$. If the bandwidth or I/O resource are not sufficient at some time slots, the corresponding scheduled checkpoint event is canceled. The bandwidth B_i and I/O O_i are allocated to maximize the number of task i's checkpoint events, while minimizing $T_{n,i}$ and $T_{b,i}$ is the second objective (i.e., if there is a tie, the values of B_i and I/O O_i are selected to minimize $T_{n,i}$ and $T_{b,i}$, respectively). A modified Dijkstra algorithm is employed to find maximum flow with bandwidth B_i.

1) A centralized checkpointing scheme. There is a centralized storage server for saving all tasks' checkpoint images. The link connecting the central storage server and core switches has a capacity of $C_s = 100$ Gbps [43].
2) A peer-to-peer checkpointing scheme. All links have capacity $C_l = 10$ Gbps.

We have shown that problem JCSR is NP hard; now in order to show how far our algorithm's performance deviates from the optimal solution, we have compared reliability from our algorithm with the assumptions described in the optimal solution of Case 1, and the optimal solution of Case

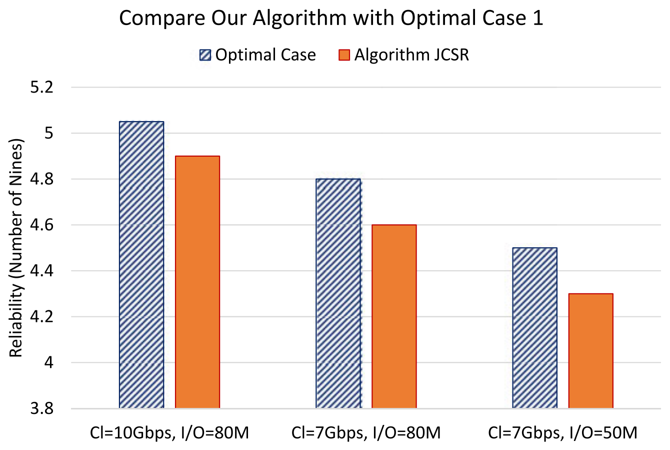


Fig. 5. Comparison of reliability in optimal solution and our JCSR algorithm of optimal Case 1, with 300 tasks running, when varying link and I/O capacity. Our algorithm has an error percentage less than 4 percent on average compared to optimal solution, and the error percentage decreases even more as resource capacity increases.

1 through the bipartite graph. Reliability is measured by the number-of-nines.[2] The simulation has 300 tasks running for the two algorithms, and we decrease link capacity and I/O capacity respectively. Fig. 5 shows the comparison of reliability from the two algorithms. Even when link capacity and I/O capacity become very limited, our algorithm has an error percentage of less than 4 percent on average compared to the optimal solution, and this error percentage decreases even more as resource capacity increases. This means that our JCSR algorithm can obtain near-optimal solutions. The runtime (on a machine with Intel(R) Core(TM) i7-3537U CPU@2.00 GHz 2.5 GHz, and 8 GB memory) and number of iterations for different numbers of tasks are listed in Table 2. Even on such a vanilla machine, the runtime is much smaller than the typical checkpointing interval, such as $\{40,000, 50,000\}$ seconds as reported in [41], [42]. We would also like to point out that in practice, the checkpoint schedule only needs to be updated at a much larger timescale than the checkpoint interval itself, because the same scheduling/routing decisions will be used during the execution of a given set of tasks (often consisting of many checkpoint intervals).

Impact of Checkpoint Scheme. In Section 3 we have introduced the peer-to-peer checkpointing mechanism to avoid the bottleneck of link capacity in centralized checkpointing. Now we will show how this peer-to-peer checkpointing outperforms centralized mechanism in the view of reliability. Again we have 300 tasks running for each checkpoint mechanism, centralized scheme has a link capacity (between central storage server and the switch) of 100 Gbps, peer-to-peer checkpointing has bandwidth capacity of 10 Gbps on each link, in both mechanism each node has a I/O capacity of 80 MB/s. Fig. 6 shows the cumulative distributed function (cdf) of reliability, for the two baseline schemes and our proposed reliability optimization algorithm. From which we can see that peer-to-peer checkpointing with random parameters outperforms centralized scheme when multiple

2. Reliability R_i can be equivalently measured by the number-of-nines, i.e., $-\log_{10}(1 - R_i)$. For instance, four nines correspond to a reliability of 99.99 percent.

TABLE 2
JCRS Algorithm Run Time

Number of Tasks	Runtime	Iterations
100	1768.94	12
200	3136.6	13
300	5499.57	13
400	5967.07	14
500	8278.25	13

tasks are running, i.e., number of nines in reliability increases from 3.25 to 3.75 on average, this is because peer-to-peer checkpointing utilizes higher bandwidth by distributing checkpoint traffic over all links. And our proposed algorithm with peer-to-peer checkpointing shows even more significant reliability improvement: it improves reliability by roughly one order of magnitude over the centralized scheme, from 99.9 percent (i.e., three nines) to 99.99 percent (i.e., four nines). Such an improvement is due to the coordination of checkpoint traffics, which becomes nearly orthogonal in temporal or spatial domain.

Impact of Link Capacity. Problem JCSR has two capacity constraints, based on that checkpoint times consists of T_n, which depends on I/O resources on the host, and T_b, depending on link bandwidth. We'll study how these constraints affect our solution. First we go with link capacity, fix I/O capacity at 80 MB/s at each node, decreasing bottleneck link capacity C_s by 30 and 60 percent in centralized checkpointing mechanism, and decreasing each link capacity C_l by 30 and 60 percent in peer-to-peer checkpointing, 300 tasks running. Scaling down all link capacity expectantly reduces reliability, because it causes higher congestion in the network. Fig. 7 shows that our proposed algorithm with peer-to-peer checkpointing outperforms the centralized scheme even when bottleneck link capacity is $C_s = 100$ Gbps and peer-to-peer scheme only has an link capacity of 4 Gbps. Peer-to-peer checkpointing and our algorithm for joint

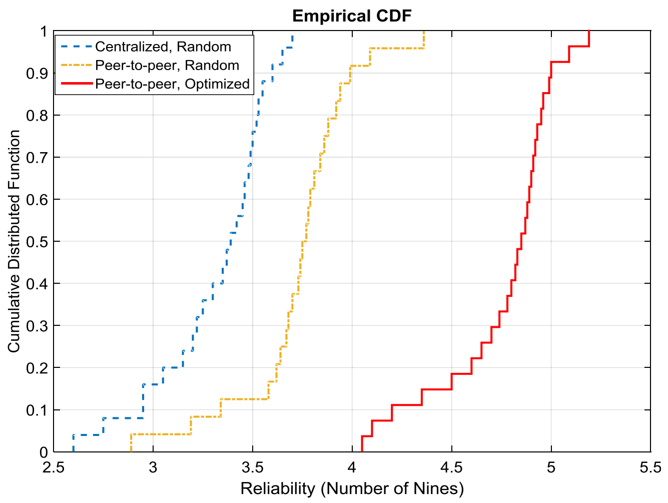


Fig. 6. Comparison of joint reliability of 300 tasks on centralized/Fat-tree topology. Centralized scheme has a bandwidth capacity at the switch of 100 Gbps, distributed checkpointing has link capacity of 10 Gbps on each link. Peer-to-peer checkpointing outperforms centralized scheme when multiple jobs are included. And our proposed algorithm with peer-to-peer checkpointing shows even more significant reliability improvement.

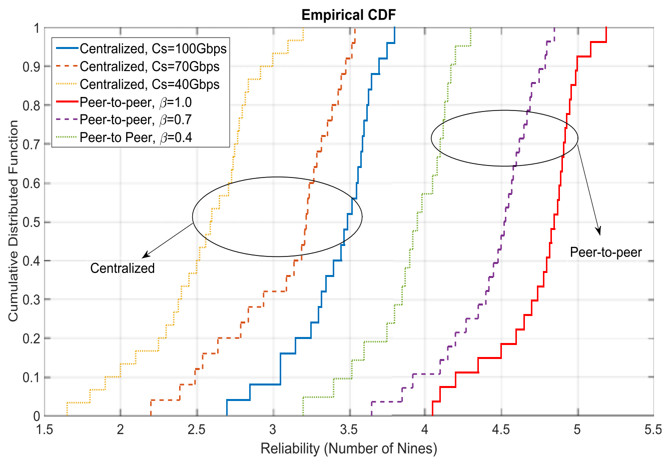


Fig. 7. Impact of changing link capacity: fix I/O capacity at 80 MB at each node, decreasing bandwidth capacity (C_s and C_l) by 30 and 60 percent in both centralized/distributed typologies. Reliability decreases as bandwidth capacity decreases in both cases. Our proposed algorithm with peer-to-peer checkpointing outperforms the centralized scheme even when bottleneck link capacity is $C_s = 100$ Gbps and peer-to-peer scheme only has an link capacity 60 percent off 10 Gbps (4 Gbps).

checkpoint scheduling and routing provide a link-cost-effective solution for achieving reliability. It mitigates the cost of deploying high capacity links in data centers.

Impact of I/O Capacity. Next, we will study the impact of I/O capacity on reliability with the two checkpointing schemes (centralized and peer-to-peer). Now we fix link capacity $C_s = 100$ Gbps in centralized scheme and $C_l = 10$ Gbps in peer-to-peer checkpointing. Varying I/O capacity at each node from 50 MB/s to 100 MB/s. In Fig. 8 we can see that reliability decreases as available I/O capacity decreases in both checkpointing mechanisms, since with the same number of tasks running, lower I/O capacity means more congestion in saving checkpoint image, which increases service downtime. The figure also shows that our proposed algorithm with peer-to-peer checkpointing outperforms the centralized scheme even when centralized I/O has a capacity of 100 MB/s while peer-to-peer only has 50 MB/s at each node.

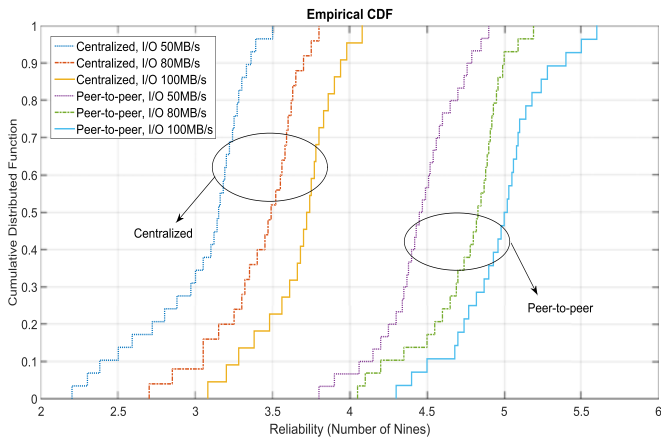


Fig. 8. Impact of changing I/O capacity: fix link capacity $C_s = 100$Gbps in centralized scheme and $C_l = 10$Gbps in peer-to-peer checkpointing. Varying I/O capacity at each node. Reliability decreases as I/O capacity decreases in both topologies. Our proposed algorithm with peer-to-peer checkpointing outperforms the centralized scheme even when centralized I/O has a capacity of 100 MB/s while peer-to-peer only has 50 MB/s at each node.

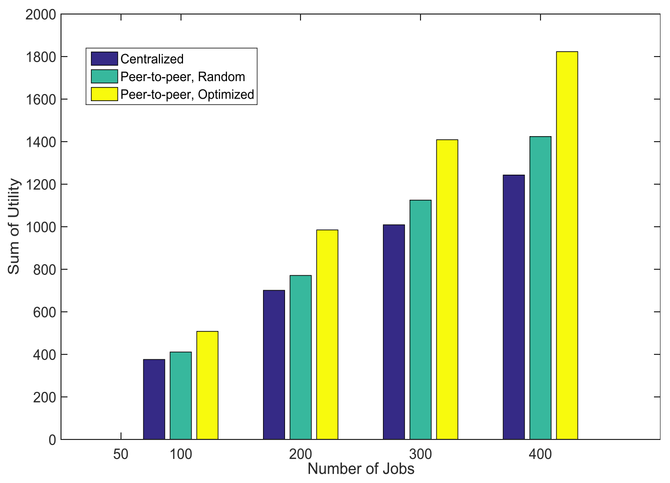


Fig. 9. Sum of utility of all tasks. Different number of tasks running: {100, 200, 300, 400} in the three checkpointing schemes. Our proposed algorithm outperforms the other two in reliability, and the improvement is more significant as number of tasks increases.

This validates that peer-to-peer checkpointing and our JCSR algorithm can also mitigates the cost of I/O resources on host machines, which provides another I/O-cost-effective solution for achieving reliability as well.

Impact of Task Size. The number of tasks running in algorithm JCSR also affects the solution on reliability, we show the sum of utility of all tasks running in the three checkpointing mechanisms in Fig. 9, where link capacity for peer-to-peer checkpointing is 10 Gbps and that for centralized checkpointing is $C_s = 100$ Gbps, I/O capacity at each node is 80 MB/s, we calculate the sum of utility as a function of reliability: $U_i(R_i) = -w_i \log_{10}(1 - R_i)$, with $w_i = 1$, i.e., utility of task i yields to number of nines in reliability. The number of tasks running in algorithm JCSR ranges from the set {100, 200, 300, 400}. Fig. 9 shows that sum of utility increases as number of tasks increases, which is intuitive. However, we can also see from the figure that in centralized checkpointing scheme, this increase is less than linear compared to the other two peer-to-peer checkpointing schemes, this is because as number of tasks increases, congestion at the bottleneck link will be more severe, which leads to reliability loss. The increase in number of tasks running has the least impact on our optimized peer-to-peer checkpointing, whose sum utility increase is roughly linear as shown in the figure as the number of tasks increases linearly. This also validates that our JCSR algorithm with joint checkpoint scheduling and routing is scalable to large tasks sets.

6 Conclusion and Future Work

This paper proposes a novel approach to providing elastic reliability optimization in cloud computing. Relying on peer-to-peer checkpointing, the problem of joint reliability maximization is formulated as an optimization, in which data center operators need to find checkpoint scheduling and make routing/placement decisions in order to maximize an aggregate utility of reliability. The resulting optimization problem, which is shown to be non-convex and combinatorial, is efficiently solved using a distributed algorithm based on dual decomposition. Numerical examples with synthesized traffic trace shows that our solution significantly improves reliability by an order of magnitude over both random peer-to-peer and centralized checkpointing mechanisms.

In ongoing work we are looking at providing reliability maximization under dynamic job arrivals and departures, as well as for non-Poisson failure models. We are also working on reliability optimization algorithms which not only allow time-varying checkpoint scheduling (e.g., non-deterministic checkpoint intervals), but also incorporate dynamic routing/placement algorithms. We hope that the results presented in this paper provide fuel to understanding and prototyping reliability services in cloud computing.

ACKNOWLEDGMENTS

The authors would like to thank anonymous reviewers for their helpful suggestions, and Nakharin Limrungsi for many helpful discussions on initial simulation setup. This work is supported in part by US National Science Foundation grant 1320226, and CAREER Award CNS-1350766.

REFERENCES

[1] Amazon, "We promise our EC2 cloud will only crash once a week," Amazon, Seattle, Washington, USA, Tech. Rep., Oct. 2008, http://www.businessinsider.com/2008/10/amazon-we-promise-our-ec2-cloud-will-only-crash-once-a-week-amzn-

[2] RackSpace, "Software as a service perceptions survey," RackSpace, Windcrest, TX, USA, Tech. Rep. Mar. 2007. [Online]. Available: www.rackspace.com/downloads/surveys/SaaSSurvey.pdf

[3] VMware, "Protecting mission-critical workloads with VMware fault tolerance," Tech. Rep., [Online]. Available: www.vmware.com/files/pdf/resources/ft_virtualization_wp.pdf, Feb. 2009.

[4] P. Ta-Shma, G. Laden, M. Ben-Yehuda, and M. Factor, "Virtual machine time travel using continuous data protection and checkpointing,"*ACM SIGOPS Operating Syst. Rev.*, vol. 42, pp. 127–134, 2008.

[5] A. Warfield, R. Ross, K. Fraser, C. Limpach, and S. Hand, "Parallax: Managing storage for a million machines," in *Proc. 10th Workshop Hot Topics Operating Syst.*, Jun. 2005, pp. 4–4.

[6] R. Badrinath, R. Krishnakumar, and R. Rajan, "Virtualization aware job schedulers for checkpoint-restart," in*Proc. 13th Int. Conf. Parallel Distrib. Syst.*, Dec. 2007, pp. 1–7.

[7] I. Goiri, F. Juli'a, J. Guitart, and J. Torres, "Checkpoint-based fault-tolerant infrastructure for virtualized service providers," in *Proc. IEEE/IFIP Netw. Operations Manage. Symp.*, Aug. 2010, pp. 455–462.

[8] M. Zhang, H. Jin, X. Shi, and S. Wu, "VirtCFT: A transparent VM-level fault-tolerant system for virtual clusters," in *Proc. Int. Conf. Parallel Distrib. Syst.*, Dec. 2010, pp. 147–154.

[9] Y. Liu, R. Nassar, C. Leangsuksun, N. Naksinehaboon, M. Paun, and S. L. Scott, "An optimal checkpoint/restart model for a large scale high performance computing system," in *Proc. Int. Symp. Parallel Distrib. Process.*, Apr. 2008, pp. 1–9.

[10] S. Yi, D. Kondo, and A. Andrzejak, "Reducing costs of spot instances via checkpointing in the amazon elastic compute cloud," in *Proc. IEEE 3rd Int. Conf. Cloud Comput.*, Jul. 2010, pp. 236–243.

[11] S. kandula, J. padhye, and V. bahl, "Flyways to De-Congest Data Center Networks," in *8th ACM Workshop Hot Topics Networks.*, Oct. 2009, http://research.microsoft.com/apps/pubs/default.aspx?id=136452.

[12] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge Univ. Press, 2005.

[13] R. Miller, "When The Power Goes Out at Google," 2010. [Online]. Available: http://www.datacenterknowledge.com/archives/2010/03/08/when-the-power-goes-out-at-google/

[14] P. Thibodeau, "Amazon cloud outage was triggered by configuration error," 2011, http://www.computerworld.com/article/2508335/cloud-computing/amazon-cloud-outage-was-triggered-by-configuration-error.html

[15] Z. Guo, et al., "Failure recovery: When the cure is worse than the disease," in *Proc. 14th Workshop Hot Topics Operating Syst.*, 2013, p. 8.

[16] X. Zhou and T. Nishizeki, "Edge-coloring and f-coloring for various classes of graphs," in *Proc. 5th Int. Symp. AlgorithmsComput.*, 1994, pp. 199–207.

[17] S. Nakano, X. Zhou, and T. Nishizeki, "Edge-coloring algorithms," in *Computer Science Today: Recent Trends and Developments*, Ed. Jan Van Leeuwen, Berlin, Germany: Springer-Verlag, 1995, pp. 172–183.

[18] R. Mysore, A. Pamboris, and A. Vahdat, "Portland: A scalable fault-tolerant layer 2 data center network fabric," in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2009, pp. 39–50.

[19] D. Ersoz, M. S. Yousif, and C. R. Das, "Characterizing network traffic in a cluster-based, multi-tier data center," in *Proc. 27th Int. Conf. Distrib. Computi. Syst.*, 2007, pp. 59–69.

[20] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C.R. Das, "Towards characterizing cloud backend workloads: Insights from Google compute clusters," *SIGMETRICS Perform. Eval. Rev.*, vol. 37, pp. 34–41, Mar. 2010.

[21] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 92–99, Jan. 2010.

[22] E. G. Coffman, M. R. Garey, D. S. Johnson, and A. S. LaPaugh, "Scheduling file transfers in a distributed network," in *Proc. 2nd Annu. ACM Symp. Principles Distrib. Comput.*, 1983, pp. 254–266.

[23] Y. Cai and M. C. Kong, "Nonpreemptive scheduling periodic tasks uni-multiprocessor systems," *Algorithmica*, vol. 15, no. 6, pp. 572–599, Jun. 1996.

[24] A. Chowdhury and P. Tripathi, "Enhancing cloud computing reliability using efficient scheduling by providing reliability as a service," in *Proc. Int. Conf. Parallel, Distrib. Grid Comput.*, Dec. 2014, pp. 99–104.

[25] A. Zhou, S. Wang, Z. Zheng, C. Hsu, M. Lyu, and F. Yang, "On cloud service reliability enhancement with optimal resource usage," *IEEE Trans. Cloud Comput.*, Doi: 10.1109/TCC.2014.2369421.

[26] D. Tiwari, S. Gupta, and S.S. Vazhkudai, "Lazy checkpointing: exploiting temporal locality in failures to mitigate checkpointing overheads on extreme-scale systems," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2014, pp. 25–36.

[27] Rahul Singh, David Irwin, Prashant Shenoy, and K. K. Ramakrishnan, "Yank: Enabling green data centers to pull the plug," in *Proc. 10th USENIX Symp. Netw. Syst. Des. Implementation*, Apr. 2013, pp. 142–156.

[28] S. Ren and M. A. Islam, "Colocation demand response: Why do i turn off my servers?," presented at the 11th Int. Conf. Autonomic Computing, Philadelphia, PA, USA, Jun. 2014.

[29] C. Stewart, A. Chakrabarti, and R. Grifth, "Zoolander: Efficiently meeting very strict, low-latency SLOs," presented at the 10th Int. Conf. Autonomic Computing, San Jose, CA, USA, Jun. 2013.

[30] K. Gardner, S. Zbarsky, E. Hyytia, and A. Scheller-Wolf, "Reducing latency via redundant requests: Exact analysis," *SIGMETRICS '15*, Jun. 2015, New York, NY, USA.

[31] B. Mills, R. E. Grant, K. B. Ferreira, and R. Riesen, "Evaluating energy savings for checkpoint/restart," in *Proc. 1st Int. Workshop Energy Efficient Supercomput.*, 2013, Art. no. 6.

[32] P. Hoenisch, S. Schulte and S. Dustdar, "Workflow scheduling and resource allocation for cloud-based execution of elastic processes," in *Proc. IEEE 6th Int. Conf. Serv.-Oriented Comput. Appl.*, 2013, pp. 1–8.

[33] Z. Wu, X. Liu, Z. Ni, D. Yuan and Y. Yang, "A market-oriented hierarchical scheduling strategy in cloud workflow systems," *J. Supercomputing*, vol. 63, no. 1, pp. 256–293, Jan. 2013.

[34] S. W. Kwak, B. J. Choi and B. K. Kim, "An optimal checkpointing-strategy for real-time control systems under transient faults," *IEEE Trans. Rel.*, vol. 50, no. 3, pp. 293–301, Sep. 2001.

[35] Zhang, Y. and K. Chakrabarty, "Energy-aware adaptive checkpointing in embedded real-time systems," in *Proc. Des., Autom. Test Eur. Conf. Exhib.*, 2003, p. 10918.

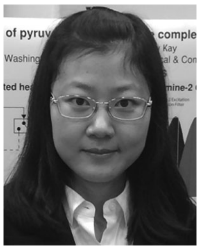
[36] S. Yi, D. Kondo, B. Kim, G. Park, and Y. Cho, "Using replication and checkpointing for reliable task management in computational Grids," in *Proc. Int. Conf. High Perform. Comput. Simul.*, Jun. 2010, pp. 125–131.

[37] H. R. Faragardi and R. Shojaee, "An analytical model to evaluate reliability of cloud computing systems in the presence of QoS requirements," in *Proc. IEEE/ACIS 12th Int. Conf. Comput. Inform. Sci.*, Jun. 2013, pp. 315–321.

[38] S. Fu and C. Z. Xu, "Proactive resource management for failure resilient high performance computing clusters", in *Proc. Int. Conf. Availability, Rel. Security*, 2009, pp. 257–264.

[39] R. S. Matos, Jr, P. R. M. Maciel, F. Machida, D. S. Kim and K. S. Trivedi, "Sensitivity analysis of server virtualized system availability," *IEEE Trans. Rel.*, vol. 61, no. 4, pp. 994–1006, Dec. 2012.
[40] D. Lorenzoli and G. Spanoudakis, "Predicting software service availability: Towards runtime monitoring approach," in *Proc. IEEE Int. Conf. Web Serv.*, 2011, pp. 736–737.
[41] Prabhat and Quincey Koziol, *High Performance Parallel I/O*. London, U.K: Chapman and Hall, Oct. 2014, pp. 284–286.
[42] T. Wang, W. Yu, S. Oral, B. W. Settlemyer, and S. Atchley, "An efficient distributed burst buffer for Linux," in *Proc. Lustre User Group Conf.*, Miami, United States, 2014.
[43] Cisco Catalyst 4948 Switch Data Sheet. (2014, Jan.). [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-4900-series-switches
[44] D. Xu, Y. Li, M. Chiang, and A. R. Calderbank, "Elastic service availability: Utility framework optimal provisioning," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 6, pp. 55–65, Aug. 2008.

Juzi Zhao received the MS and PhD degrees from the Department of Electrical and Computer Engineering, The George Washington University in 2009 and 2014, respectively. She is with the Department of Electrical and Computer Engineering at University of Massachusetts-Lowell. Her current research interests include optical and data center networks.



Yu Xiang received the BASc degree from Harbin Institute of Technology in 2010, and the PhD degree from George Washington University in 2015, both in electrical engineering. She is now a senior inventive scientist at AT&T Labs-Research. Her current research interests include cloud resource optimization, distributed storage systems, and cloud storage charge-back.

Tian Lan received the BASc degree from the Tsinghua University, China, in 2003, the MASc degree from the University of Toronto, Canada, in 2005, and the PhD degree from the Princeton University in 2010. He is currently an assistant professor of electrical and computer engineering at the George Washington University. His research interests include cloud resource optimization, distributed systems, and cyber security. He received the 2008 IEEE Signal Processing Society Best Paper Award, the 2009 IEEE GLOBECOM Best Paper Award, and the 2012 INFOCOM Best Paper Award. He is a member of the IEEE.

H. Howie Huang received the PhD in computer science from the University of Virginia. He is an associate professor in the Department of Electrical and Computer Engineering, George Washington University. His research interests include the areas of computer systems and architecture, including cloud computing, big data, and high-performance computing. He was a recipient of the NSF CAREER award, NVIDIA Academic Partnership Award, and IBM Real Time Innovation Faculty Award. He is a senior member of the IEEE.

Suresh Subramaniam received the PhD degree in electrical engineering from the University of Washington, Seattle, in 1997. He is a professor in the Department of Electrical and Computer Engineering, George Washington University, Washington, DC. His research interests include the architectural, algorithmic, and performance aspects of communication networks, with current emphasis on optical networks, cloud computing, and data center networks. He has published more than 150 peer-reviewed papers in these areas, and has co-edited 3 books on optical networking. He has served as TPC chair for several conferences including Globecom 2006 and 2016 ONS, LANMAN 2014, INFOCOM 2013, ANTS 2008, and ICC 2007 ONS. He is or has been on the editorial boards of seven journals including the *IEEE/ACM Transactions on Networking* and the *IEEE/OSA Journal of Optical Communications and Networking*. He is a recipient of Best Paper Awards at ICC 2006 ONS and at the 1997 SPIE Conference on All-Optical Communication Systems. He is a fellow of the IEEE.