Optimizing Job Reliability Through Contention-Free, Distributed Checkpoint Scheduling

Yu Xiang, Hang Liu, Tian Lan, Howie Huang, Suresh Subramaniam
Department of ECE, George Washington University

Abstract

A datacenter that consists of hundreds or thousands of servers can provide virtualized environments to a large number of cloud applications and jobs that value the requirement of reliability very differently. Checkpointing a virtual machine (VM) is a proven technique to improve reliability. However, existing checkpoint scheduling techniques for enhancing reliability of distributed systems have fallen short, either because they tend to offer the same, fixed reliability to all jobs, or because their solutions are tied up to specific applications and rely on centralized checkpoint control mechanisms. In this work, we first show that reliability can be significantly improved through contention-free scheduling of checkpoints. Then, inspired by the Carrier Sense Multiple Access (CSMA) protocol in wireless congestion control, we propose a novel framework for distributed and contention-free scheduling of VM checkpointing to provide reliability as a transparent, elastic service. We quantify reliability in closed form by studying system stationary behaviours, and maximize job reliability through utility optimization. Our design is validated via a proof-of-concept prototype that leverages readily available implementations in Xen hypervisors. The proposed checkpoint scheduling is shown to significantly reduce checkpointing interference and improve reliability by as much as one order of magnitude over contention-oblivious checkpoint schemes.

1 Introduction

Reliability is a critical requirement for modern datacenters. Clearly, high reliability is desirable for many jobs and applications, because even a small service downtime may potentially lead to business interruption with hefty financial penalties. In a public cloud, reliability is provided as a fixed service parameter, e.g., all Amazon EC2 users are expected to receive 99.95% reliability [1]. In other words, the best reliability that cloud customers can get right now is also the worst. Thus it is up to the cloud customers to harden the jobs running within their VM instances in order to enhance reliability for critical applications. Now, applications that provide increased reliability *do* exist, e.g., Oracle's payroll and general ledger programs. However, these approaches are specific to applications or require specific problem structures, and providing elastic reliability for the masses remains an elusive goal in cloud computing today.

This paper introduces an approach for assigning elastic reliability to heterogeneous datacenter jobs via distributed checkpoint scheduling and reliability optimization. Virtual Machine (VM) checkpointing is a widely-employed, application-transparent solution to improve reliability in public clouds [4, 5]. To optimize reliability of *a single job*, prior work has proposed a number of models for calculating the optimal checkpoint schedule [6, 7, 8, 9, 10, 11], and several algorithms for balancing checkpoint workload and performance overhead have also been proposed in [14, 15, 16]. Unfortunately, these solutions fall short in optimizing checkpoints of *multiple jobs* whose reliability requirements may vary significantly, due to their inadequacy of taking into account resource contention among different jobs' checkpoints.

In a multi-job scenario, uncoordinated VM checkpoints taken independently run the risk of interfering with each other [12, 13] and may cause significant resource contention and reliability degradation [2]. In particular, the time to save local checkpoint images is determined largely by how I/O resources are shared, while the overhead to transfer locally saved images to networked storage relies on how network resources are shared. In a large datacenter, chances that VM checkpointing, if unmanaged and uncoordinated, would encounter severe network and I/O congestion, resulting in high VM checkpointing overhead and reliability loss. Clearly, for a large datacenter, a centralized checkpoint scheduling scheme that micro-manages each job's checkpoints is impracti-

cal for handling tens of thousands of jobs. Distributed checkpoint scheduling is needed for achieving our goal of providing elastic reliability as a service.

To this end, we propose a novel job-level self-management approach that not only enables distributed checkpoint scheduling but also optimizes reliability assignments to individual jobs. Our contention-free scheduling solution is inspired by the Carrier Sense Multiple Access (CSMA) method, a distributed protocol for accessing a shared transmission medium, wherein a node verifies the absence of other traffic before transmitting on the medium [24, 25, 26]. If a job senses any ongoing checkpoint actions at its serving hosts, it waits (or backs-off) for an indefinite amount of time and keeps silent if any of its hosts is busy or become busy during its backoff. We compare our method with contention-oblivious checkpoint scheduling, wherein each job simply checkpoints its VMs at a predetermined rate regardless of any contention from other jobs's checkpoint. To the best of our knowledge, this is the first work using a CSMA-based scheme for distributed datacenter resource scheduling and reliability optimization. The main contributions of this paper are summarized as follows:

- We harness CSMA-based interference management to provide a distributed and contention-free checkpoint scheduling protocol. Our solution is well suited for implementation in large-scale datacenters, as its job-level distributed checkpoint scheduling mechanism can effectively mitigate resource contentions caused by concurrent job checkpoints.
- Reliability received by each individual job is characterized in closed form by studying the stationary behavior of our proposed protocol. It enables a joint reliability optimization where flexible service-level agreements (SLAs) are negotiated through a joint assessment of all jobs' reliability requirements and total datacenter resources available.
- Results are validated via a proof-of-concept prototype that leverages readily available implementations in Xen and Linux. The proposed CSMA-based checkpoint scheduling is shown to significantly reduce checkpoint interference and improve reliability by 1 order of magnitude over contention-oblivious checkpoint schemes.

The rest of the paper is organized as follows. Section II introduces the system model and illustrates the necessity for distributed, contention-free checkpoint scheduling. Our theoretical analysis of CSMA-based checkpoint scheduling and reliability optimization are presented in Section III. Section IV contains experimental results via a proof-of-concept prototype, and Section V concludes the paper.

2 System Model and Motivations

2.1 Reliability Model Using Checkpoints

In the simplest form, a Virtual Machine Monitor (VMM) can periodically record a clear state of the running VMs, including a full image of the VM's memory, CPU, and all the device states, and flush resulting VM images to a central storage server to establish recovery points [12, 17, 18]. For a single job consisting of multiple VMs, uncoordinated checkpoints taken independently of each other [12, 13] run the risk of cascaded rollbacks if causality is not respected. This can be avoided by taking synchronous checkpoints of all the VMs that a job comprises, whereas the probability of checkpoint contention increases as jobs often consist of multiple VMs.

As shown in Figure 1, a single job periodically checkpoints all its VMs every T_i seconds. Each checkpoint requires time $T_n + T_f$ to complete, which includes time T_n to suspend all its VM executions in order to save a local checkpoint image, as well as time T_f to transfer the saved VM images to a remote destination. After a failure occurs, the job can be restored from an available checkpoint and rolled back to the last saved state with recovery time T_r. We define reliability by 1 minus the fraction of expected service downtime. More precisely, let a failure occur at time t after the nth checkpoint is fully completed. Then,

$$\begin{aligned} R &= 1 - \mathbb{E}\left[\frac{\text{Service Downtime}}{\text{Total Service Time}}\right] \\ &= 1 - \mathbb{E}\left[\frac{t-(n-1)T_i - T_n - T_f + nT_n + T_r}{t+T_r}\right] \end{aligned} \quad (1)$$

where nT_n is the total service downtime due to taking checkpoints, $t-(n-1)T_i - T_n - T_f$ is the lost service time due to roll-back, and $\mathbb{E}$ is an expectation over all random factors, e.g., checkpoint overhead and failure time. For clarification, we summarize main notations in this paper in Table 1.

For a single job, a number of proposals for determining the optimal temporal scheduling of checkpoints have been provided in [14, 15, 16, 19, 20]. Further, protocols for taking consistent snapshots of distributed services in virtualized environments using OpenFlow hardware to improve fault tolerance are presented in [21]. However, these results do not take into account the contention among different checkpoints in a multi-job scenario. It is shown that as the size of datacenter grows large, uncoordinated VM checkpoints from different jobs may cause significant resource contention and result in high checkpoint overhead T_n (due to I/O resource contention) and T_f (due to network resource contention) [2], which directly translate to severe reliability degradation according to (1). Recently, a utility-based framework for joint

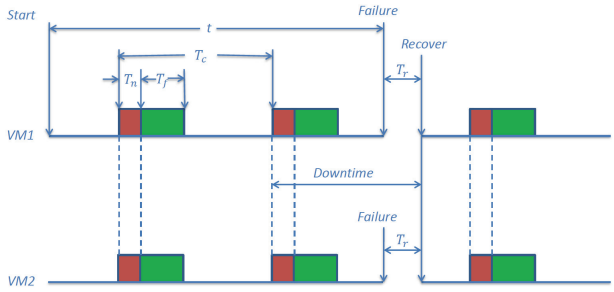


Figure 1: Multiple VMs belonging to the same job must be checkpointed simultaneously to avoid cascaded rollbacks. It increases the chance of checkpoint contention.

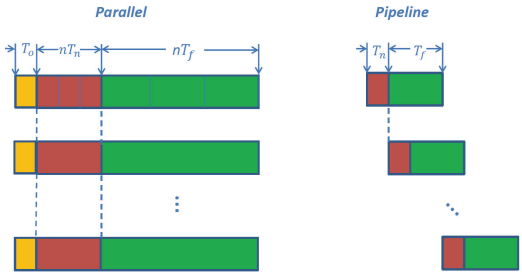


Figure 2: Fully coordinated checkpoint scheduling in a pipeline mode significantly reduces resource contention over parallel checkpoints.

reliability maximization under datacenter resource constraints is proposed in [2]. It is shown to reduce expected service downtime by as much as an order of magnitude, even though the solution requires centralized coordination/scheduling and does not allow a distributed implementation at a large scale.

2.2 Need for Contention-Free Checkpoint Scheduling

Clearly, job reliability benefits from mitigating checkpoint overhead, while checkpoint frequency also has to be determined to amortize not only service downtime due to taking checkpoints but also potential service loss due to failure and roll-back. Due to resource sharing in datacenters, all of these require a joint checkpoint scheduling over all jobs that share a common pool of resources. Consider two extreme cases for multi-job checkpoint scheduling: parallel and pipeline scheduling, as illustrated in Figure 2. In parallel mode, the checkpoints of all N jobs are done at the same time and the total I/O and network bandwidth are shared among them. In theory, the time to save a local checkpoint T_n and to transfer VM images T_f will be at least N times higher than when checkpoints are taken one at a time, and there can also be an overhead T_o for switching between VM checkpoints. On the other hand, if fine-grained checkpoint control is possible, checkpoints of jobs can be taken one immediately after another in a pipelined fashion by overlapping image-saving time of one job's checkpoint with the image transfer time of another job. With such completely coordinated checkpoints, jobs can take full advantage of all I/O and network bandwidth resource available, causing minimal interference to others.

Table 1: Main notation.

Symbol	Meaning
$\mathcal{N}$	N job indexed by $i = 1,\ldots,N$
$\mathcal{S}$	S hosts indexed by $h = 1,\ldots,S$
λ_i	Sensing rate of job i
μ_i	Service rate to checkpoint job i
τ_i^c	Mean checkpoint time of job i
τ_i^r	Mean rollback and recovery time of job i
f_i	Mean failure rate of job i
R_i	Reliability of job i
$\mathcal{X}_k$	A state in our Markov Chain model
$P_{\mathcal{X}_k,\mathcal{X}_l}$	Transition rate between states $\mathcal{X}_k$ and $\mathcal{X}_l$
π_k	Stationary distribution in state $\mathcal{X}_k$
$\mathcal{A}_i$	A set of all states containing job i
$\mathbb{E}[Y]$	Expectation of random a variable Y

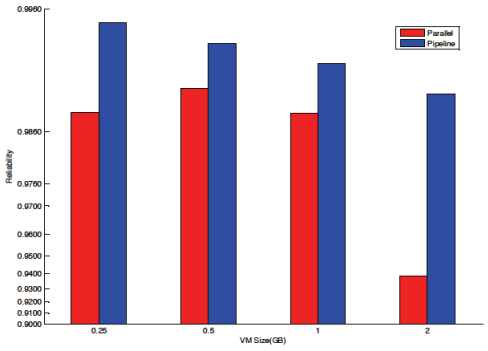


Figure 3: Fully coordinated pipeline checkpoint schedule significantly reduces contention and improves reliability over parallel checkpoint schedule. Reliability calculated with 8 failures/year.

To demonstrate the advantage of checkpoint coordination, we set up a simple experiment involving two hosts and four VMs on each host to quantify how much reliability is achieved under each scheme. We implement both parallel and pipeline scheduling, and measure the checkpoint overhead and VM image transfer time. Figure 3 shows that pipeline scheduling outperforms parallel

scheduling by nearly an order of magnitude for various VM sizes. Further, the reliability improvement increases as VM size grows because the larger the VM size, the longer time it needs to save and transfer checkpoint images, and more likely checkpoint contentions occur.

We conclude that checkpoint scheduling is crucial to provide high reliability in a multi-job scenario. Even though pipeline scheduling completely avoids checkpoint interference and is able to efficiently utilize all I/O and network bandwidth available, such a centralized coordination and micro-management approach is prohibitive in large-scale datacenters hosting tens of thousands of jobs. Therefore, a practical checkpoint scheduling scheme should (i) allow a distributed implementation without relying on any centralized, fine-grained checkpoint control, (ii) be able to schedule contention-free checkpoints for a large number of jobs that may have heterogeneous parameters and demands, and (iii) enable a joint reliability maximization to assign the optimal reliability level to each job that suits its demand. To this end, this paper makes novel use of the CSMA protocol in wireless inference control to derive a distributed, contention-free checkpoint scheduling protocol with joint reliability optimization.

3 Our Proposed Protocol and Reliability Optimization

In this section, we propose a CSMA-based checkpoint scheduling protocol and quantify the resulting reliability received by each job in closed-form. Unlike existing work [24, 25, 26] that apply CSMA to wireless interference management and are often concerned with data throughput, our reliability analysis quantifies the reliability received by each job in closed-form and enables joint reliability optimization of all jobs via a utility framework. The protocol is inspired by CSMA but is applied to datacenter resource sharing to achieve distributed, contention-free checkpoint scheduling protocol in large-scale datacenters. Figure 4 shows an overview of the system architecture. Each job may consist of one or more VMs, which are distributed to different physical machines (or servers). Our checkpoints are organized at job level - if a checkpoint of a job is triggered, all VMs that belong to the job first save their checkpoint images to local storage (in order to minimize VM downtime) and then transfer them to networked storage to avoid host failure.

In our design, *each job achieves reliability optimization via self-management in two ways*: first, each job autonomously determines its own checkpointing scheduling based on locally available information, e.g., the colocation of other jobs and occurrence of checkpoint contention. Second, each job autonomously updates its checkpoint rate based on locally available optimal solutions, which is done in runtime with no dependence on any centralized management decisions.

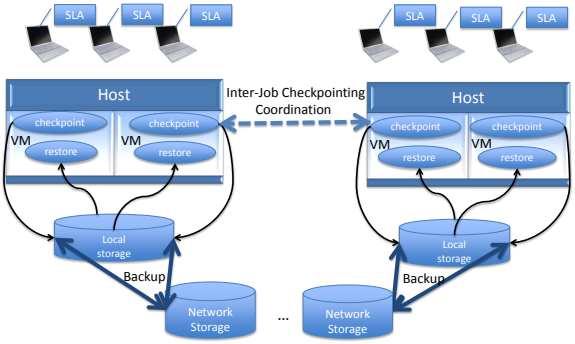


Figure 4: Our proposed architecture for checkpoint scheduling.

In this section, we will first introduce the CSMA-based checkpoint scheduling protocol, characterize the resulting reliability via a Markov Chain analysis of system stationary distributions, and then present a joint reliability optimization. Theoretical results obtained in this section will be validated through a prototype implementation in Section IV.

3.1 CSMA-Based Checkpoint Scheduling

We consider a datacenter serving N jobs denoted by $\mathcal{N} = \{1,2,\ldots,N\}$ and using S servers denoted by $\mathcal{S} = \{1,2,\ldots,S\}$. Each job i is comprised of h_i VMs that are hosted on a subset of servers, i.e., $\mathcal{H}_i \subseteq \mathcal{S}$.

As discussed in Section II, mitigating checkpoint contention can significantly reduce service downtime and improve reliability. To develop a checkpoint scheduling protocol that is not only contention-free but also fully distributed, we extend an idealized model of CSMA as in [24, 25, 26]. CSMA is a probabilistic medium access control protocol in which a node verifies the absence of other traffic before transmitting on a shared transmission medium. Our proposed checkpoint scheduling works as follows: Each job i makes the decision to create a remote checkpoint image based only on its local parameters and observation of contention. If job i senses ongoing checkpoints at any of its serving hosts (i.e., any host s such that $s \in \mathcal{H}_i$), then it keeps silent. If none of its serving hosts is busy, then job i waits (or backs-off) for a random period of time which is exponentially distributed with mean $1/\lambda_i$ and then starts its checkpointing.[1] During the back-off, if some contending job starts taking checkpoints, then job i suspends its back-off

[1]The random backoff time is to ensure that two potentially-contending jobs that sense no contention from other jobs do not start checkpointing at the same time and trigger a contention.

and resumes it after the contending checkpoint is complete. For analytical tractability, we assume that the total time of saving a local checkpoint and transferring it to a remote destination is exponentially distributed with mean $1/\mu_i = \mathbb{E}(T_n + T_f)$. This assumption of exponential checkpoint time can be further removed using results in [26]. In such an idealized CSMA model, if sensing time is negligible and back-off time follows a continuous distribution, then the probability for two contending checkpoints to start at the same time is 0 [24]. Therefore, the CSMA-based protocol, summarized in Figure. 5 achieves contention-free, distributed scheduling of job checkpoints.

```
Assign positive sensing rates λ_i > 0 ∀i

Each job independently performs:
Initialize backoff timer B_i
while job i is running
  while B_i > 0
    if any server in H_i is busy
      Job i keeps silent
      Generate new backoff: B_i = exponential with mean 1/λ_i
    end if
    Update B_i = B_i − 1
  end if
  Checkpoint all VMs of job i
  Generate new backoff: B_i = exponential with mean 1/λ_i
end while
```

Figure 5: Our contention-free, distributed checkpoint scheduling protocol inspired by CSMA.

In contrast to existing CSMA analysis that focuses on data throughput, this paper aims to quantify individual-job reliability resulting from such contention-free, distributed checkpoint scheduling protocol in large-scale datacenters. It requires us to investigate the distributions checkpoint interval T_i which is a random variable due to the CSMA-based, probabilistic checkpoint scheduling protocol. Given a set of sensing rates $\lambda_1, \ldots, \lambda_N$, we use R_i to denote the reliability received by job i in the proposed checkpoint scheduling protocol. Notice that reliability also depends on service rates $\mu_1, \ldots, \mu_N$ and job failure rate f_i, which may further depend on server failure model and VM placement. In this paper, we focus on the checkpoint scheduling protocol and reliability maximization by optimizing parameters $\lambda_1, \ldots, \lambda_N$. Using our model, we are able to find the reliability received by each job in closed form and then perform reliability optimization jointly over all jobs with respect to their utilities.

3.2 Markov Chain Model

In order to optimize reliability, we first need to obtain the reliability each job receives in the CSMA-based checkpoint scheduling protocol for given sensing rates $\lambda_1, \ldots, \lambda_N$. We make use of a Markov Chain model, which is commonly employed for CSMA analysis in wireless interference management. The Markov Chain for analyzing the protocol depends on sensing rate λ_i and checkpoint overhead μ_i, as well as datacenter VM placement that determines the pattern of job interference. We will first describe the model in this subsection and then use it to derive job reliability in closed form to enable reliability optimization.

For any time t, we define a system state as the set of jobs actively taking checkpoints at t. Since our CSMA-based protocol achieves contention-free checkpoint scheduling, in each state, a set of non-conflicting jobs (known as an Independent Set) are scheduled. We assume that there exist $K \leq 2^N$ possible states, represented by $\mathscr{X}_k \subseteq \mathscr{N}$, for $k = 1, \ldots, K$. In state $\mathscr{X}_k$, if job i is not taking checkpoints and all of its conflicting jobs are not taking checkpoints, the state $\mathscr{X}_k$ can transit to state $\mathscr{X}_k \cup \{i\}$ with a rate λ_i (i.e., job i starts its checkpoint). Similarly, state $\mathscr{X}_k \cup \{i\}$ can transit to state $\mathscr{X}_k$ with a rate μ_i (i.e., job i completes its checkpoints). It is easy to see that the system state at any time is a Continuous Time Markov Chain (CTMC).

Unlike existing CSMA analyses for wireless systems that focus on throughput, our goal is to quantify job reliability using the Markov Chain model. According to (1), this requires the characterization of the distribution of checkpoint overhead T_n, T_f, and checkpoint interval T_i, which are related to sojourn time and returning time of the CTMC. We first transform the CTMC into an embedded Discrete Time Markov Chain (DTMC) that is easier to analyze. Since the embedded chain also has different holding times for its states, we further apply the uniformization technique to obtain a randomized DTMC. It is sufficient to consider transitions between states that differ by one job because there is no contention in our idealized CSMA model. Let v be a uniformization constant that is sufficiently large. Then, the DTMC has the following transition probabilities:

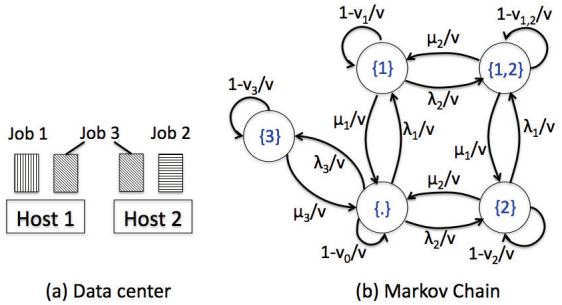
$$P_{\mathscr{X}_k, \mathscr{X}_k \cup \{i\}} = \frac{\lambda_i}{v} \text{ and } P_{\mathscr{X}_k \cup \{i\}, \mathscr{X}_k} = \frac{\mu_i}{v}, \quad (2)$$

where $P_{\mathscr{X}_k, \mathscr{X}_l}$ denote the transition probabilities from state $\mathscr{X}_k$ to state $\mathscr{X}_l$. Due to uniformization, we define $v_k = \sum_{l \neq k} v \cdot P_{\mathscr{X}_k, \mathscr{X}_l}$ to be the sum of transition probabilities out of state $\mathscr{X}_k$ and add a self-transition rate $1 - v_k/v$ so that the transition probabilities form a stochastic matrix. This means we have

$$P_{\mathscr{X}_k, \mathscr{X}_k} = 1 - \frac{v_k}{v}. \quad (3)$$

Now we can study properties of the original CTMC through the DTMC whose state transitions occur according to the jump times of an independent Poisson Process

with rate v. Fig. 6 (a) gives an example data center with 3 jobs and 2 hosts. If each host is able to checkpoint 1 VM at a time without incurring any performance loss, then checkpoints of job 3 conflicts with those of jobs 1 and 2, whereas jobs 1 and 2 can take parallel checkpoints without any resource contention. Therefore, this system has $K = 5$ feasible states (or Independent Sets): $\{\cdot\}$, $\{1\}$, $\{2\}$, $\{3\}$, $\{1,2\}$. State $\{\cdot\}$ means no job is taking checkpoints, $\{i\}$ means a single job i takes checkpoints for $i = 1,2,3$, and $\{1,2\}$ means jobs 1 and 2 take checkpoints at the same time.



(a) Data center (b) Markov Chain

Figure 6: Example: 3 jobs and corresponding Markov Chain.

Given the above DTMC model, we are interested in analyzing its stationary behavior, which reveals the distributions of checkpoint overhead T_n, T_f, and checkpoint interval T_i. The transition probability matrix P of the DTMC has size K by K and its stationary distribution is denoted by $\pi_1, \ldots, \pi_K$, satisfying

$$(\pi_1, \ldots, \pi_K) = (\pi_1, \ldots, \pi_K) \cdot P, \tag{4}$$

where π_k is the stationary probability that the DTMC stays in state $\mathscr{X}_k$. In the following lemma, we show that the stationary distribution can be obtained in closed form for our DTMC model.

Lemma 1 *When no checkpoint interference (i.e., contention) is permitted, the DTMC has stationary distribution:*

$$\pi_k = \frac{\prod_{i \in \mathscr{X}_k} \lambda_i \cdot \prod_{j \notin \mathscr{X}_k} \mu_j}{C_\lambda}, \tag{5}$$

where C_λ is a normalization factor such that $\sum_k \pi_k = 1$.

Proof: This lemma can be directly proved by showing that the stationary distribution in (5) satisfies the detailed balance equation $\pi_k P_{\mathscr{X}_k, \mathscr{X}_l} = \pi_l P_{\mathscr{X}_l, \mathscr{X}_k}$, $\forall k, l$. Therefore, the DTMC is time-reversible and its stationary distribution depends on rates λ_i, μ_i of all jobs. □

3.3 Reliability Analysis

Now we can analyze the stationary behavior of the CTMC through the DTMC and an independent Poisson Process with rate v. From (1), reliability is defined by 1 minus the percentage of service downtime. It means that we need to obtain the distributions of checkpoint overhead T_n, T_f, and checkpoint interval T_i from the Markov Chain model. We assume that each job has known Mean Time to Failure (MTTF) $1/f_i$ and its failure time is modeled by an exponential distribution. In practice, the MTTF can be estimated from existing failure models or large-scale datacenter event logs [27, 28]. For example, if each server has independent failures according to a Poisson Process with rate f_0 and job i is hosted by m_i different servers, then we have $f_i = m_i \cdot f_0$.

Consider checkpoint overhead T_n, T_f, and checkpoint interval T_i in our CSMA-based protocol for a single job i. Let $\mathscr{A}_i = \{\mathscr{X}_k : i \in \mathscr{X}_k\}$ be a set of all states containing job i. It is not hard to see that total checkpoint overhead $T_n + T_f$ is the sojourn time that the CTMC stays within $\mathscr{A}_i$, i.e., the time to checkpoint job i's VMs. Similarly, checkpoint interval T_i is the first returning time of the CTMC to $\mathscr{A}_i$. Clearly, both sojourn time and first returning time are random variables whose distributions depend on the Markov Chain model. Using the definition in (1), we first rewrite reliability R_i with respect to random checkpoint overhead and checkpoint interval.

Lemma 2 *Let T_i be the random checkpoint interval of job i. If job i has Poisson failures with rate f_i, then its reliability is given by*

$$R_i = 1 - \tau_i^c \mu_i \pi_{\mathscr{A}_i} - f_i \pi_{\mathscr{A}_i} \mathbb{E}T_i - f_i \tau_i^r - \frac{f_i \mathbb{E}\left(T_i^2\right)}{2\mathbb{E}T_i} \tag{6}$$

where τ_i^c is the mean time to save a local checkpoint image, τ_i^r is mean repair time, and $\pi_{\mathscr{A}_i} = \sum_{k \in \mathscr{A}_i} \pi_k$ is the sum of stationary distribution of all states in $\mathscr{A}_i$.

The result is very intuitive. First, $\pi_{\mathscr{A}_i}$ is the fraction of time that the Markov Chain spends in states $\mathscr{A}_i$ (i.e., checkpointing job i VMs). Since our protocol is contention-free, out of $\mathbb{E}(T_n + T_c) = 1/\mu_i$ seconds on average for each checkpoint, job i VMs have to be suspended for $\mathbb{E}(T_n) = \tau_i^c$ seconds to save consistent, local checkpoint images. Therefore, the service downtime due to taking checkpoints is given by $\tau_i^c \mu_i \pi_{\mathscr{A}_i}$. Second, $f_i \tau_i^r$ is the expected downtime due to failure recovery and repair. Further, because of our assumption of Poisson failures, lost service time due to VM roll-back after each failure can be derived using the Poisson Arrival Sees Time Average (PASTA) property, i.e, $f_i \mathbb{E}\left(T_i^2\right)/2\mathbb{E}T_i$. Finally, when a failure arrives before a checkpoint is completed (which again has probability $\pi_{\mathscr{A}_i}$), all VMs must be recovered from the last available checkpoint images.

It implies that an additional roll-back time of $\pi_{\mathscr{A}_i}\mathbb{E}T_i$ is incurred on average. A formal proof for this lemma can be found in our online technical report [32].

Therefore, reliability of job i can be obtained if $\mathbb{E}T_i$ and $\mathbb{E}T_i^2$ are known. Next we derive them via their counterparts in the embedded DTMC. Since job i takes a checkpoint if the DTMC is in a state belonging to $\mathscr{A}_i$, its checkpoint interval T_i can be measured by the first returning time to $\mathscr{A}_i$, denoted by $t_{\mathscr{A}_i}$. Let $Y_1, Y_2 \ldots$ be a sequence of i.i.d. exponentially-distributed variables with mean $1/v$. We have $T_i = \sum_{l=1}^{t_{\mathscr{A}_i}} Y_l$, which results in

$$\mathbb{E}T_i = \mathbb{E}t_{\mathscr{A}_i} \cdot \mathbb{E}Y_l = \frac{1}{v}\mathbb{E}t_{\mathscr{A}_i} \tag{7}$$

and

$$\begin{aligned}\mathbb{E}T_i^2 &= \mathrm{var}(Y_l) \cdot \mathbb{E}t_{\mathscr{A}_i} + (\mathbb{E}Y_l)^2 \cdot \mathbb{E}t_{\mathscr{A}_i}^2 \\ &= \frac{1}{v^2}\left(\mathbb{E}t_{\mathscr{A}_i} + \mathbb{E}t_{\mathscr{A}_i}^2\right). \end{aligned} \tag{8}$$

Here we used the i.i.d. property of Y_l, as well as the independence between the DTMC and the underlying Poisson Process.

Now it remains to find $\mathbb{E}t_{\mathscr{A}_i}$ and $\mathbb{E}t_{\mathscr{A}_i}^2$ in the embedded DTMC. When the number of jobs is large, we can approximate the first returning time $t_{\mathscr{A}_i}$ by an exponential distribution [29]. Then, its second order moment should be $\mathbb{E}t_{\mathscr{A}_i}^2 = 2\mathbb{E}t_{\mathscr{A}_i}^2$. To find $\mathbb{E}t_{\mathscr{A}_i}$, we apply Kac's Formula in [29] and obtain the following result:

Lemma 3 *The expectation of first returning time $t_{\mathscr{A}_i}$ for the DTMC is given by*

$$\mathbb{E}t_{\mathscr{A}_i} = 1 + \frac{v}{\mu_i}\left(\frac{1}{\pi_{\mathscr{A}_i}} - 1\right). \tag{9}$$

Proof: Checkpoint interval $t_{\mathscr{A}_i}$ is the time that the DTMC first returns to any state in $\mathscr{A}_i$ since it last left. Let $\mathscr{X}_{(n)}$ be the DTMC state at time n under stationary distribution. It is easy to see that

$$t_{\mathscr{A}_i} = \min\{T | \mathscr{X}_{(0)} \in \mathscr{A}_i, \mathscr{X}_{(1)} \notin \mathscr{A}_i, \mathscr{X}_{(T)} \in \mathscr{A}_i\}, \tag{10}$$

that is the minimum (random) time the chain returns to $\mathscr{A}_i$ after it leaves at time $n = 0$. Applying Kac's Formula [29] to the DTMC, we have $1/\pi_{\mathscr{A}_i} = \mathbb{E}\left[\tau^+_{\mathscr{A}_i}\right]$, where $\tau^+_{\mathscr{A}_i} = \min\{T | \mathscr{X}_{(0)} \in \mathscr{A}_i, \mathscr{X}_{(T)} \in \mathscr{A}_i, T \geq 1\}$ is the first hitting time from a stationary distribution. Using the Law of total probability, we further have

$$\begin{aligned}&\mathbb{E}\left[\tau^+_{\mathscr{A}_i}\right] \\ &= \frac{v-\mu_i}{v}\mathbb{E}\left[\tau^+_{\mathscr{A}_i} | \mathscr{X}_{(1)} \in \mathscr{A}_i\right] + \frac{\mu_i}{v}\mathbb{E}\left[\tau^+_{\mathscr{A}_i} | \mathscr{X}_{(1)} \notin \mathscr{A}_i\right], \\ &= \frac{v-\mu_i}{v} + \frac{\mu_i}{v}\mathbb{E}\left[t_{\mathscr{A}_i}\right], \end{aligned} \tag{11}$$

where the first step uses $\mathbb{P}_{\{\mathscr{X}_{(1)} \in \mathscr{A}_i\}} = 1 - \mu_i/v$ and $\mathbb{P}_{\{\mathscr{X}_{(1)} \notin \mathscr{A}_i\}} = \mu_i/v$ because departure probability from $\mathscr{A}_i$ is a constant μ_i/v from all states. The second step uses the definition of $t_{\mathscr{A}_i}$ in (10), as well as the fact that $\mathbb{E}\left[\tau^+_{\mathscr{A}_i} | \mathscr{X}_{(1)} \in \mathscr{A}_i\right] = 1$ due to the definition of hitting time. Combining (11) and Kac's formula $1/\pi_{\mathscr{A}_i} = \mathbb{E}\left[\tau^+_{\mathscr{A}_i}\right]$, we derive the desired equation in (9). This completes the proof. □

Plugging these results into (6), we can quantify the reliability received by each job i in our contention-free, distributed checkpoint scheduling protocol. Again, we refer reader to our online technical report [32] for a complete proof.

Theorem 1 *For given rates $\lambda_1, \ldots, \lambda_K$, each job i in our protocol receives the following reliability R_i:*

$$R_i = 1 - f_i\tau_i^r - \tau_i^c\mu_i\pi_{\mathscr{A}_i} - \frac{f_i}{\mu_i}\left(\pi_{\mathscr{A}_i} + \frac{1}{\pi_{\mathscr{A}_i}}\right) \tag{12}$$

3.4 Reliability Optimization

We can use Theorem 1 to numerically calculate the reliability of each job i for any given rates $\lambda_1, \ldots, \lambda_K$ and failure rate f_i. Let $U_i(R_i)$ be a utility function, representing the value of assigning reliability level r_i to job i. Our goal is to derive an autonomous reliability optimization where flexible SLAs are negotiated through a joint assessment of users utility and total datacenter resources available. Toward this end, we formulate a joint reliability optimization through a utility optimization framework [30, 31] that maximizes total utility $\sum_i U_i(R_i)$, i.e.,

$$\begin{aligned}\max \quad & \sum_i U_i(R_i) \\ \text{s.t.} \quad & R_i = 1 - f_i\tau_i^r - \tau_i^c\mu_i\pi_{\mathscr{A}_i} - \frac{f_i}{\mu_i}\left(\pi_{\mathscr{A}_i} + \frac{1}{\pi_{\mathscr{A}_i}}\right), \\ & \pi_{\mathscr{A}_i} = \frac{1}{C_\lambda} \cdot \sum_{\mathscr{X}_k \in \mathscr{A}_i} \prod_{j \in \mathscr{X}_k} \lambda_j \cdot \prod_{l \notin \mathscr{X}_k} \mu_l, \\ \text{var.} \quad & \lambda_1, \ldots, \lambda_K \end{aligned} \tag{13}$$

where C_λ is a normalization factor such that $\sum_k \pi_k = 1$. Here we used the closed-form reliability characterization in (12) and the stationary distribution in (5).

The reliability optimization is computed by maximizing an aggregate utility $\sum_i U_i(R_i)$ over all feasible sensing rates $\lambda_1, \ldots, \lambda_K$. In a dynamic setting, such reliability optimizations must be solved for each job arrival and departure to balance reliability assignment autonomously. We notice that many local search heuristics, such as Hill Climbing [33] and Simulated Annealing [34], can be employed to solve the reliability optimization in (13) by incrementally improving the total utility over single search

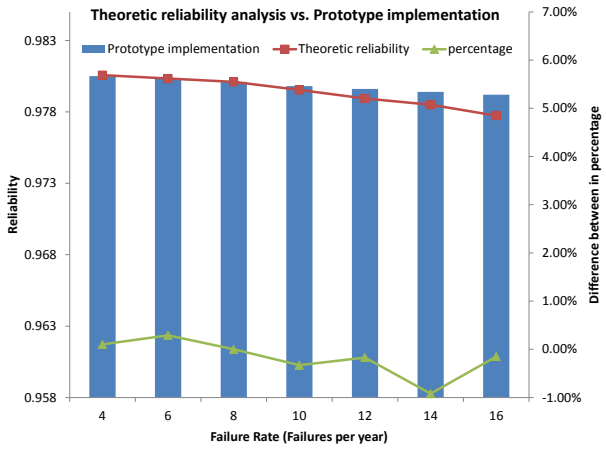


Figure 7: Comparison of the reliability values from our theoretical analysis with a prototype experiment using 24 VMs in Xen. Our reliability analysis can accurately estimate reliability in the proposed contention-free checkpoint scheduling protocol within a margin of $\pm 1\%$.

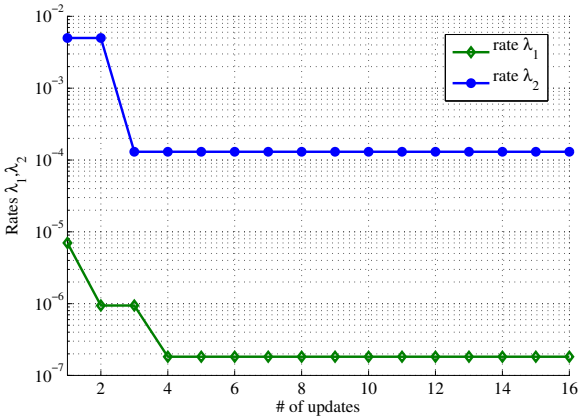


Figure 8: Plot convergence of sensing rates λ_1, λ_2 when Hill Climbing local search [33] is employed to solve the reliability optimization in (13) with 2 classes of jobs and a utility $2R_1 + R_2$. The algorithm converges within only a few local updates to the optimal sensing rates.

directions. Under certain conditions, we can also characterize the optimal solution in closed form.

Theorem 2 *If there exists a set of rates $\lambda_1, \ldots, \lambda_K$ and a positive constant C_λ satisfying the following system of equations, then the rates maximize the aggregate utility in (13) for arbitrary non-decreasing functions:*

$$\begin{aligned} \textstyle\sum_{\mathscr{X}_k \in \mathscr{A}_i} \prod_{j \in \mathscr{X}_k} \lambda_j \cdot \prod_{l \notin \mathscr{X}_k} \mu_l = C_\lambda \cdot \sqrt{\frac{\tau_i^c \mu_i^2}{f_i} + 1}, \forall i \\ \textstyle\sum_{k=1}^{K} \prod_{j \in \mathscr{X}_k} \lambda_j \cdot \prod_{l \notin \mathscr{X}_k} \mu_l = C_\lambda \end{aligned} \tag{14}$$

These rates simultaneously maximize the reliabilities received by all jobs, i.e.,

$$R_i = 1 - f_i \tau_i^r - 2\sqrt{\tau_i^c + \frac{f_i}{\mu_i}^2}, \ \forall i. \tag{15}$$

Proof: We apply the following inequality, $ax + b/x \geq 2\sqrt{ab}$ for all positive $a, b, x > 0$, to the reliability in (12). It implies

$$\begin{aligned} R_i &= 1 - f_i \tau_i^r - \tau_i^c \mu_i \pi_{\mathscr{A}_i} - \frac{f_i}{\mu_i}(\pi_{\mathscr{A}_i} + \frac{1}{\pi_{\mathscr{A}_i}}) \\ &\leq 1 - f_i \tau_i^r - 2\sqrt{\tau_i^c + \frac{f_i}{\mu_i}^2}, \end{aligned} \tag{16}$$

where we used $x = \pi_{\mathscr{A}_i}$, $a = \tau_i^c \mu_i + \frac{f_i}{\mu_i}$ and $b = \frac{f_i}{\mu_i}$ in the inequality. Notice that the last step holds with equality only if $x = \sqrt{b/a}$. For arbitrary non-decreasing utility functions $U_i(R_i)$, it is easy to see that aggregate utility $\sum_i U_i(R_i)$ is maximized if (16) holds with equality for all $i = 1, \ldots, N$, i.e., all reliability values are maximized simultaneously. This proves the maximum achievable reliability in (15), which can be achieved only if $\pi_{\mathscr{A}_i} = \sqrt{\frac{\tau_i^c \mu_i^2}{f_i} + 1}$, $\forall i$. Plugging the stationary distribution in (5), this is exactly conditions (14). □

Remark: Theorem 2 establishes the maximum utility our checkpoint scheduling algorithm can achieve. If the conditions in Theorem 2 are satisfied, then solving (14) gives us a set of rates $\lambda_1, \ldots, \lambda_K$, which maximize the aggregate utility in (13) for arbitrary non-decreasing utility functions. As an example, if all jobs share a common resource bottleneck that allows only a single checkpoint at each time, then we have $\mathscr{A}_i = \{i\}$ $\forall i$ because any pair of jobs conflict with each other. It is easy to verify that the following rates satisfy conditions (14), and therefore the reliability optimization can be solved in closed form for arbitrary non-decreasing utility functions:

$$\lambda_i = \frac{\sqrt{\frac{\tau_i^c \mu_i^2}{f_i} + 1}}{\sum_{j=1}^{N} \sqrt{\frac{\tau_j^c \mu_j^2}{f_j} + 1}} \cdot \frac{1 - \prod_{j=1}^{N} \mu_j}{\sum_{j=1}^{N} \prod_{l \neq j} \mu_l}, \ \forall i. \tag{17}$$

Once the optimal solutions are obtained (through either local search heuristics or the sufficient conditions above), each job only has to update its checkpoint rate according to the optimal solutions. Due to the distributed nature of CSMA-based scheduling, jobs can easily reconfigure their checkpoint rates on-the-fly without relying on any centralized checkpoint coordination.

Validation of theoretical analysis. To validate the

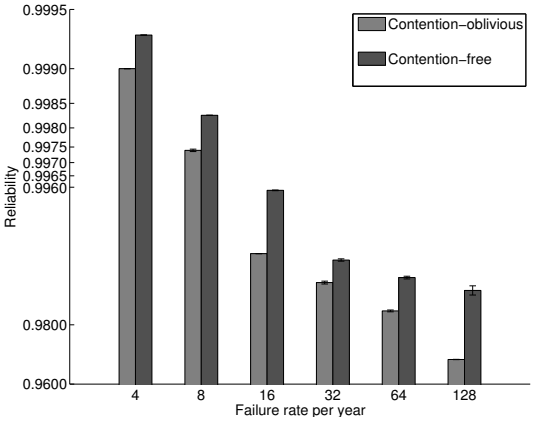


Figure 9: Reliability for different failure rates.

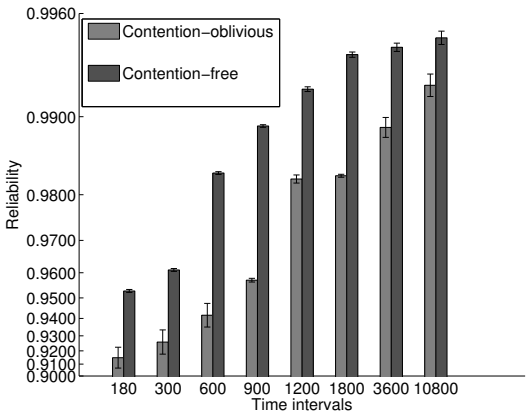


Figure 10: Reliability for different checkpoint time intervals.

reliability analysis in Theorem 1, we implement a prototype of the contention-free, distributed checkpoint scheduling protocol with 3 servers supporting 24 Xen VMs each with 1GB DRAM. The detailed implementation parameters are provided later in Section IV. We first benchmark necessary parameters in our theoretical model using Markov Chain analysis, i.e., mean checkpoint local-saving time $\tau_i^c = 30.2$ seconds, mean checkpoint overhead $1/\mu_i = 71.5$ seconds, and mean repair time $\tau_i^r = 80.2$ seconds for all jobs $i = 1,\ldots,24$. So all jobs in the experiment receive equal reliability value. For a sensing rate of $\lambda_i = 1/(2.5$ days) and exponential failures with f_i ranging from 2 to 16 failures per year, we compare the reliability values from our theoretical analysis to the values obtained from the experiment. Figure 7 shows that our theoretical analysis can accurately estimate the reliability values received in the proposed protocol, with a small error margin of $\pm 1\%$. This implies that our theoretical reliability analysis provides a powerful tool for reliability estimation and optimization.

Example for reliability optimization. To give a numerical example of the proposed reliability optimization, consider a data center with 2 classes of jobs: 10 large jobs that contain 10 VMs each and 100 small jobs that contain 2 VMs each. Assume that at most 2 jobs can take non-contending checkpoints at each time. Average checkpoint overhead is $\tau_1^c = 50$ seconds for large jobs and $\tau_2^c = 25$ for small jobs. Recovery time is $\tau_1^r = 400$ seconds and $\tau_2^r = 200$ seconds. Assume that each host has independent failures with rate f_0 =2/year. Then, large jobs have failure rates $f_1 = 10 \cdot f_0 = 6.43e-7$ and small jobs $f_2 = 2 \cdot f_0 = 1.29e-7$. Finally, total checkpoint time is $1/\mu_{\text{large}} = 200$ seconds for a large job and $1/\mu_{\text{large}} = 100$ seconds for a small job. We implement Hill Climbing local search [33] to find the optimal sensing rates λ_1, λ_2 that maximize a utility $2R_1 + R_2$. As shown in Figure 8, the algorithm converges within a few local updates to the optimal sensing rates. At optimum, large jobs receive a higher reliability $R_1 = 0.99$ than small jobs $R = 0.90$ because the weight of large jobs is twice as that of small jobs in the optimization objective $2R_1 + R_2$.

4 Implementation and Evaluations

We have implemented a prototype of the contention-free checkpoint scheduling in C. We use a cluster of four machines with Intel Atom CPU D525, 4GB DRAM, 7200 RPM 1TB hard drives, and interconnected with a 1GB/s network. Note that I/O and network bandwidths rather than CPU and memory are the major limiting factors for our tests. To simulate the workload, each VM runs a CPU intensive benchmark [35] with 1 VCPU, 512MB or 1GB DRAM, and 10GB VDisk. The host OS is Linux 2.6.32 and Xen 4.0. If not specified, the failure rate is eight times per year, and each reliability result is the average of three runs.

Figure 9 shows the reliability of a job when the annual failure rate varies from 4 times to 128 times per year. In this experiment, we run three jobs (two VMs per job, and six VMs in total) and present the average reliability. For small failure rates, the reliabilities for both contention-oblivious and contention-free scheduling are very high. But as more failures occur, the benefit of contention-free scheduling becomes very obvious, achieving much higher reliability.

Reliability as a function of checkpoint interval is shown in Figure 10. Overall, contention-free scheduling can achieve a reliability of two nines ($> 99\%$), compared

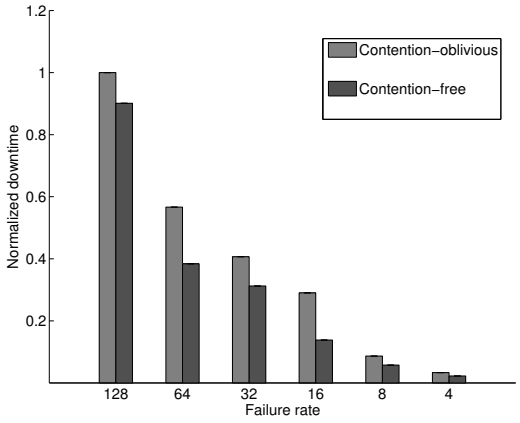


Figure 12: Normalized downtime for different annual failure rates.

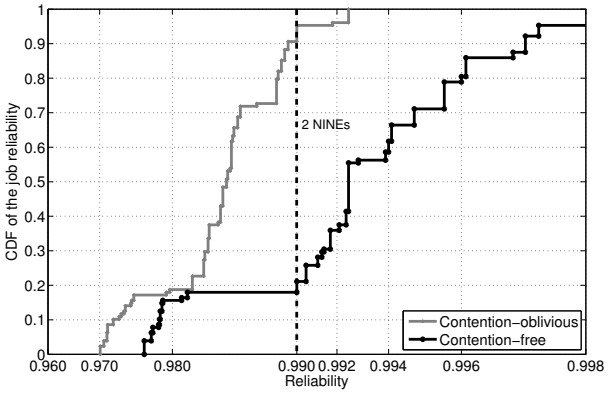


Figure 11: Reliability of 128 jobs for both contention-free and contention-oblivious checkpoint scheduling.

to one nine ($> 90\%$) for contention-oblivious scheduling. For contention-free scheduling, the reliability of the system keeps increasing as the checkpoint interval becomes larger. At the same time, the contention-oblivious mechanism increases at a slower pace, but it can also potentially reach as high reliability as contention-free scheduling. This happens because when the checkpoint interval becomes large enough, chances for checkpoint contention from different jobs are small. To demonstrate the scale of our approach, we also extend this test to simulating 128 jobs. In this experiment, we intentionally intensify the job checkpointing rate in our cluster. As shown in Figure 11, almost all contention-free configuration jobs can achieve a reliability of two nines but the major percentage of contention-oblivious jobs falls into one nine reliability range. In addition, we present the normalized downtime for different annual failure rate settings in Figure 12. Note that the downtime of a system includes the checkpoint time, and recovery time if the host is down. All times are normalized to the downtime for contention-oblivious scheduling with 128 failures per

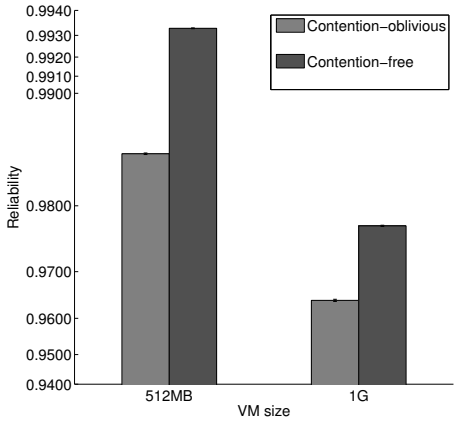


Figure 13: Reliability for different VM sizes.

year. One can see that our contention-free checkpointing can achieve a reduction in downtime of upto 18.3% compared to contention-oblivious scheduling.

In Figure 13, we show the effect of VM memory size (and, therefore, checkpoint duration) on reliability. Under the same checkpoint interval and failure rate, a job with VM memory size of 512 MB achieves higher reliability due to its smaller memory footprint. The reason is that a larger DRAM size requires more time to suspend the VM and transfer the VM image from the host machine to the destination storage. But the overall trend still shows that our contention-free checkpointing mechanism significantly outperforms contention-oblivious scheduling.

5 Conclusions

Inspired by the CSMA protocol for wireless interference management, we propose a new protocol for distributed and contention-free checkpoint scheduling in large-scale datacenters, where jobs' requirements for reliability vary significantly. The protocol enables datacenter operators to provide elastic reliability as a transparent service to their customers. Using Markov Chain analysis of system stationary behaviours, the reliability that each job receives in our protocol is characterized in closed form. We also present optimization algorithms to jointly maximize all reliability levels with respect to an aggregate utility. Our design is validated through prototype implementations in Xen and Linux, and significant reliability improvements over contention-oblivious scheduling checkpointing are demonstrated via experiments in realistic settings.

6 Collection of Proofs

6.1 Proof of Lemma 1

It is easy to verify that the stationary distributions in Lemma 1 satisfies the detailed balance equation, $\pi_k P_{\mathscr{X}_k,\mathscr{X}_l} = \pi_l P_{\mathscr{X}_l,\mathscr{X}_k}$, $\forall k,l$. It also implies that the Markov chain is time reversible [29].

Stationary distribution of DTMC satisfies the detailed balance equation, $\pi_k P_{\mathscr{X}_k,\mathscr{X}_l} = \pi_l P_{\mathscr{X}_l,\mathscr{X}_k}$, $\forall k,l$, where P is the Markov transition matrix, i.e., $P_{\mathscr{X}_k,\mathscr{X}_l}$ is the probability of transition from state $\mathscr{X}_k$ to state $\mathscr{X}_l$, and π_k and π_l are the equilibrium probabilities of being in states $\mathscr{X}_k$ and $\mathscr{X}_l$, respectively. Substitute $\pi_k = \frac{\prod_{i\in\mathscr{X}_k}\lambda_i \cdot \prod_{j\notin\mathscr{X}_k}\mu_j}{C(\vec{\lambda})}$ into the detailed balanced equation as:

$$\frac{\prod_{i\in\mathscr{X}_k}\lambda_i \cdot \prod_{j\notin\mathscr{X}_k}\mu_j}{C(\vec{\lambda})} \cdot P_{\mathscr{X}_k,\mathscr{X}_l} = \frac{\prod_{i\in\mathscr{X}_l}\lambda_i \cdot \prod_{j\notin\mathscr{X}_l}\mu_j}{C(\vec{\lambda})} \cdot P_{\mathscr{X}_l,\mathscr{X}_k}$$

And jobs are either in a state or not in a state. Also since in our DTMC, adjacent states only differs in one single job, $\mathscr{X}_k$ and $\mathscr{X}_l$ only differs in one element, let us say job m, then $\prod_{i\in\mathscr{X}_k}\lambda_i$ and $\prod_{i\in\mathscr{X}_l}\lambda_i$ only differs in one element, λ_m, so is $\prod_{j\notin\mathscr{X}_k}\mu_j$ and $\prod_{j\notin\mathscr{X}_l}\mu_j$. assuming $m \in \mathscr{B} and \notin \mathscr{A}$, the lemma holds if we assume otherwise, thus we have

$$\prod_{i\in\mathscr{X}_l}\lambda_i = \lambda_m \cdot \prod_{i\in\mathscr{X}_k}\lambda_i$$

,

$$\prod_{j\notin\mathscr{X}_k}\mu_j = \mu_m \cdot \prod_{j\notin\mathscr{X}_l}\mu_j$$

then after cancellation for the equal components the left and right hand side of the equation becomes equal:

$$\mu_m \cdot \frac{\lambda_m}{v} = \lambda_m \cdot \frac{\mu_m}{v}$$

This shows the detailed balance holds for $\pi_k = \frac{\prod_{i\in\mathscr{X}_k}\lambda_i \cdot \prod_{j\notin\mathscr{X}_k}\mu_j}{C(\vec{\lambda})}$ This completes the proof.

6.2 Proof of Lemma 2

Let R be the fraction of downtime due to rollback when a failure happens, then according to PASTA, the probability that a failure arrives in any range during a checkpoint interval is the same as when it's seen as average. Then we have:

$$P(R \geq x) = \frac{\int_x^\infty (T_i - x) f_{T_i}(u)du}{\mathbb{E}T_i}$$

where $f_{T_i}(u)$ is the probability density function of check point interval T_i, let $F_R(x)$ and $f_R(x)$ be the cumulative distribution function and probability density function of downtime due to rollback R respectively, then we have Thus

$$F_R(x) = 1 - P(R \geq x) = 1 - \frac{\int_x^\infty (T_i - x) f_{T_i}(u)du}{\mathbb{E}T_i}$$

Take derivative with respect to x of both sides,

$$f_R(x) = \frac{\int_x^\infty f_{T_i}(u)du}{\mathbb{E}T_i}$$

Then the average downtime due to rollback can be expressed as:

$$\mathbb{E}R = \frac{\int_0^\infty (\int_x^\infty f_{T_i}(u)du)xdx}{\mathbb{E}T_i} = \frac{\int_0^\infty (1 - F_{T_i}(x))xdx)}{\mathbb{E}T_i},$$

where we used $\int_x^\infty f_{T_i}(u)du = 1 - \int_0^x f_{T_i}(u)du = 1 - F_{Ti}(x)$ in the last step.

$$\begin{aligned}
\mathbb{E}T_i^2 &= \int_0^\infty u^2 f_{T_i}(u)du = -\int_0^\infty u^2 d(1 - F_{T_i}(u)) \\
&= \int_0^\infty (1 - F_{T_i}(u))2udu - u^2(1 - F_{T_i}(u)) \\
&= 2\int_0^\infty (1 - F_{T_i}(u))udu - u^2(1 - F_{T_i}(u))|_0^\infty \\
&= 2\int_0^\infty (1 - F_{T_i}(u))udu
\end{aligned}$$

Since $F_{T_i}(u)$ is to 1 when u goes to infinity and $u^2(1 - F_{T_i}(u))$ goes to 0, also, when $u = 0$, $u^2(1 - F_{T_i}(u)) = 0$ as well.

It is easy to verify that the second moment of T_i can be expressed by $\mathbb{E}T_i^2 = 2\int_0^\infty (1 - F_{T_i}(u))udu$. Hence we have $\mathbb{E}R = \frac{\mathbb{E}T_i^2}{2\mathbb{E}T_i}$. Also with failure rate f_i, the average downtime due to rollback when a failure arrives is $\mathbb{E}R_{failure} = \frac{f_i \mathbb{E}T_i^2}{2\mathbb{E}T_i}$. This completes the proof.

6.3 Proof (7) and (8)

Assuming that i.i.d. random variables $\{Y_l\}$ have moment generating functions(MGF) $\phi_Y(s)$, and random sum $T_i = \sum_{l=1}^{t_{\mathscr{A}_i}} Y_l$ has MGF $\phi_T(s)$, then $\phi_T(s)$ is expect value of $\mathbb{E}[e^{sT_i}|T_i = t]$, with respect to T_i. Then we have

$$\phi_T(s) = \sum_{t=0}^{\infty} \mathbb{E}[e^{sT_i|T_i=t}]P_{T_i}(t) = \sum_{0}^{\infty} \mathbb{E}[e^{(Y_1+Y_2+\cdots+Y_t)}]P_{T_i}(t)$$

Let $W = Y_1 + Y2 + \cdots + Y_t$, then according to the definition of MGF, $\phi_W(s) = [\phi_Y(s)]^n$, which gives

$$\phi_T(s) = \sum_{t=0}^{\infty} [\phi_Y(s)]^n P_{T_i}(t)$$

in addition we can write $[\phi_Y(s)]^n = [e^{ln\phi_Y(s)}]^n = e^{[ln\phi_Y(s)]n}$, which then gives

$$\phi_T(s) = \sum_{t=0}^{\infty} e^{[ln\phi_Y(s)]n} P_{T_i}(t)$$

, where the sum on the right hand side is actually $\phi_{t_{\mathscr{A}_i}}(s)$ evaluated at $s = ln\phi_Y(s)$, therefore,

$$\phi_T(s) = \phi_{t_{\mathscr{A}_i}}(ln\phi_Y(s))$$

. Then by the chain rule of derivatives,

$$\phi'_T(s) = \phi'_{t_{\mathscr{A}_i}}(ln\phi_Y(s))\frac{\phi'_Y(s)}{\phi_Y(s)}$$

, with $\phi_Y(0) = 1$, $\phi'_Y(0) = \mathbb{E}[Y_l]$ and $\phi'_{t_{\mathscr{A}_i}}(0) = \mathbb{E}[t_{\mathscr{A}_i}]$, setting $s = 0$ the equation above becomes:

$$\mathbb{E}[T_i] = \phi'_T(0) = \phi'_{t_{\mathscr{A}_i}}(0)\frac{\phi'_Y(0)}{\phi_Y(0)} = \mathbb{E}t_{\mathscr{A}_i}\mathbb{E}[Y_l] = \frac{1}{v}\mathbb{E}t_{\mathscr{A}_i}$$

Take the second derivative of $\phi_Y(s)$, we have

$$\phi''_T(s) = \phi''_{t_{\mathscr{A}_i}}(ln\phi_Y(s))(\frac{\phi'_Y(s)}{\phi_Y(s)})^2$$

$$+\phi'_{t_{\mathscr{A}_i}}(ln\phi_Y(s))\frac{\phi_Y(s)\phi''_Y(s) - [\phi'_Y(s)]^2}{\phi^2_Y(s)}$$

Setting $s = 0$ we have

$$\mathbb{E}T_i^2 = \mathbb{E}t^2_{\mathscr{A}_i}(\mathbb{E}Y_l)^2 + Et_{\mathscr{A}_i}var(Y_l) = \frac{1}{v^2}(\mathbb{E}t_{\mathscr{A}_i}) + \mathbb{E}t^2_{\mathscr{A}_i}$$

6.4 Proof of Theorem 1

From Lemma 2:

$$R_i = 1 - \tau_i^c\mu_i\pi_{\mathscr{A}_i} - f_i\tau_i^r - (f_i\pi_{\mathscr{A}_i}\mathbb{E}T_i + \frac{f_i\mathbb{E}T_i^2}{2\mathbb{E}T_i})$$

And from Theorem 1:

$$R_i = 1 - \tau_i^c\mu_i\pi_{\mathscr{A}_i} - f_i\tau_i^r - (\frac{f_i}{\mu_i} + \frac{f_i}{2\mu_i\pi_{\mathscr{A}_i}})$$

Apply equation (7) and (8) to Lemma 2 we have:

$$f_i\pi_{\mathscr{A}_i}\mathbb{E}T_i = f_i\pi_{\mathscr{A}_i}\frac{1}{v}\mathbb{E}t_{\mathscr{A}_i}$$

$$\frac{f_i\mathbb{E}T_i^2}{2\mathbb{E}T_i} = f_i\frac{\frac{1}{v^2}(\mathbb{E}t_{\mathscr{A}_i} + \mathbb{E}t^2_{\mathscr{A}_i})}{\frac{2\mathbb{E}t_{\mathscr{A}_i}}{v}}$$

where since $t_{\mathscr{A}_i}$ is exponential distributed, its second moment

$$\mathbb{E}t^2_{\mathscr{A}_i} = 2(\mathbb{E}t_{\mathscr{A}_i})^2$$

, this gives

$$\frac{f_i\mathbb{E}T_i^2}{2\mathbb{E}T_i} = \frac{f_i}{2v}(1 + 2\mathbb{E}t_{\mathscr{A}_i})$$

in Lemma 2. Plugging in these results to Lemma 2 we have:

$$R_i = 1 - \tau_i^c\mu_i\pi_{\mathscr{A}_i} - f_i\tau_i^r - f_i(\pi_{\mathscr{A}_i}\frac{1}{v}\mathbb{E}t_{\mathscr{A}_i} + \frac{1}{2v} + \frac{1}{v}\mathbb{E}t_{\mathscr{A}_i})$$

Combining equation (9) we have:

$$\frac{1}{v}\mathbb{E}t_{\mathscr{A}_i} = \frac{1}{v} + \frac{1}{\mu_i}(\frac{1}{\pi_{\mathscr{A}_i}} - 1)$$

Apply this to Lemma 2:

$$R_i = 1 - \tau_i^c\mu_i\pi_{\mathscr{A}_i} - f_i\tau_i^r - f_i(\frac{\pi_{\mathscr{A}_i}}{v} + \frac{1-\pi_{\mathscr{A}_i}}{\mu_i} + \frac{1}{2v} + \frac{1}{v} + \frac{1}{\mu_i\pi_{\mathscr{A}_i}} - \frac{1}{\mu_i})$$

To simplify:

$$R_i = 1 - \tau_i^c\mu_i\pi_{\mathscr{A}_i} - f_i\tau_i^r - f_i(\frac{\pi_{\mathscr{A}_i}+3}{2v} - \frac{\pi_{\mathscr{A}_i}}{\mu_i} + \frac{1}{\mu_i\pi_{\mathscr{A}_i}})$$

as $v \to \infty$ we have $\frac{\pi_{\mathscr{A}_i}+3}{2v} \to 0$, thus we have:

$$R - i = 1 - f_i\tau_i^r - \tau_i^c\mu_i\pi_{\mathscr{A}_i} - \frac{f_i}{\mu_i}(\pi_{\mathscr{A}_i} + \frac{1}{\pi_{\mathscr{A}_i}})$$

This completes the proof.

References

[1] Amazon, "We Promise Our EC2 Cloud Will Only Crash Once A Week, *Amazon Online Technical Report*, October 2008

[2] N. Limrungsi, J. Zhao, Y. Xiang, T. Lan, H. Huang and S. Subramaniam, "Providing Reliability as An Elastic Service in Cloud Computing," *IEEE ICC 2012*, Aug. 2012.

[3] M. Wiboonrat, " An Empirical Study on Data Center System Failure Diagnosis ," *Internet Monitoring and Protection, ICIMP*, July. 2008.

[4] J. Hui "Checkpointing Orhestration: Toward a Scalable HPC Fault-Tolerant Environment," *CCGrid, IEEE/ACM international Symposium*, May. 2012

[5] B. Nicolae, "BlobCR: Efficient checkpoint-restart for HPC applications on IaaS clouds using virtual disk image snapshots, *2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2011

[6] N. Kobayashi and T. Dohi, "Bayesian perspective of optimal checkpoint placement," *High-Assurance Systems Engineering, 2005. HASE 2005. Ninth IEEE International Symposium on*, pp. 143-152, 2005.

[7] K.M. Chandy, "A Survey of Analytic Models of Rollback and Recovery Strategies," *Computer*, vol. 8, no. 5, pp. 40-47, May. 1975.

[8] T. Dohi, N. Kaio, and K.S. Trivedi, "Availability models with age-dependent checkpointing,", *Reliable Distributed Systems, 2002. Proceedings. 21st IEEE Symposium on*, pp. 130-139, 2002.

[9] N.H. Vaidya, "Impact of checkpoint latency on overhead ratio of a checkpointing scheme," *Computers, IEEE Transactions on*, vol. 46, no. 8, pp. 942-947, 1997.

[10] H. Okamura, Y. Nishimura and T. Dohi, "A dynamic checkpointing scheme based on reinforcement learning," *Dependable Computing, 2004. Proceedings. 10th IEEE Pacific Rim International Symposium on*, pp. 151-158, Mar. 2004.

[11] A. Duda, "The effects of checkpointing on program execution time," *Information Processing Letters*, pp. 221-229, 1983.

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

[13] M. Sun and D. M. Blough, "Fast, Lightweight Virtual Machine Checkpointing," *Georgia Tech. technical report*, 2010.

[14] I. Goiri, F. Juli'a, J. Guitart and J. Torres, "Checkpoint-based Fault-tolerant Infrastructure for Virtualized Service Providers," *in Proceedings of IEEE/IFIP Network Operations and Management Symposium*, Aug. 2010.

[15] M. Zhang, H. Jin, X. Shi, and S. Wu, "VirtCFT: A Transparent VM-Level Fault-Tolerant System for Virtual Clusters," *in Proceedings of Parallel and Distributed Systems (ICPADS)*, Dec. 2010.

[16] 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 Proceedings of Parallel and Distributed Processing (IPDPS)*, Apr. 2008.

[17] A. Warfield, R. Ross, K. Fraser, C. Limpach and S. Hand, "Parallax: Managing storage for a million machines," *HotOS*, Jun. 2005.

[18] R. Badrinath, R. Krishnakumar and R. Rajan, "Virtualization aware job schedulers for checkpoint-restart," *ICPADS*, Dec. 2007.

[19] T. Ozaki, T. Dohi, H. Okamura and N. Kaio, "Distribution-free checkpoint placement algorithms based on min-max principle," *IEEE Trans. Dependable and Secure Computing*, vol. 3, no. 2, pp. 130-140, 2006.

[20] T. Dohi, T. Ozaki, and N. Kaio, Optimal checkpoint placement with equality constraints, *Dependable, Autonomic and Secure Computing, 2nd IEEE International Symposium on*, pp. 77-84, Oct. 2006.

[21] A. Kangarlou, D. Xu, U. Kozat, P. Padala, B. Lantz and K. Igarash, "In-network Live Snapshot Service for Recovering Virtual Infrastructure," *IEEE Network*, vol. 25, no. 14, pp. 12-19, 2011.

[22] Haikun Liu, "Optimize Performance of Virtual Machine Checkpointing via Memory Exclusion," *ChinaGrid Annual Conference*, Aug 2009.

[23] B. Schroeder, E. Pinheiro, W. Weber, "DRAM errors in the wild: a large-scale field study," *in Proceeding of SIGMETRICS eleventh international joint conference on Measurement and modeling of computer systems*, pp. 193-204, 2009

[24] L. Jiangand and J. Walrand, "A distributed CSMA algorithm for throughput and utility maximization in wireless networks," *IEEE/ACM Trans. Network*, vol. 18, no. 3, pp. 960-972, Jun. 2010.

[25] X. Wang and K. Kar, Throughput Modelling and Fairness Issues in CSMA/CA Based Ad-Hoc Networks, *Proceedings of IEEE Infocom*, Miami, Mar. 2005.

[26] S.C. Liew, C. Kai, J. Leung and B. Wong, Back-of-the-Envelope Computation of Throughput Distributions in CSMA Wireless Networks, *submitted for publication http://arxiv.org//pdf/0712.1854*.

[27] International Working Group on Cloud Computing Resiliency (IWGCR), "Down-time statistics of current cloud solution," *http://iwgcr.files.wordpress.com/2012/06/iwgcr-paris-ranking-001-en1.pdf*, 2012.

[28] H. Gunawi, T. Do, J.M. Hellerstein I. Stoica, D. Borthakur and J. Robbins, "Failure as a Service (FaaS): A cloud service for large-scale, online failure drills," *http://techreports.lib.berkeley.edu/accessPages/EECS-2011-87.html*, 2012.

[29] F. Aldous, "Reversible Markov Chains and Random Walks on Graphs," *University of California, Berkeley* 2002.

[30] M. Chiang, and S.H. Low, A.R. Calderbank and J.C. Doyle, "Layering as Optimization Decomposition: A Mathematical Theory of Network Architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255-312, 2007.

[31] T. Lan, X. Lin, M. Chiang and Lee, R.B, "Stability and Benefits of Suboptimal Utility Maximization," *Networking, IEEE/ACM Transactions on*, vol. 19, no. 4, pp. 1194-1207, Aug. 2011.

[32] Y. Xiang, H.Liu, T. Lan, H. Huang, S. Subramaniam,"Optimizing Job Reliability Through Contention-Free, Distributed Checkpoint Scheduling". *Online technical report available at www.seas.gwu.edu/ tlan/papers/ICAC.pdf*, 2013.

[33] S.J. Russell, P. Norvig, "Artificial Intelligence: A Modern Approach (2nd ed.)," *Upper Saddle River, New Jersey: Prentice Hall*, pp. 111-114.

[34] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, "Optimization by Simulated Annealing," *Science 220 (4598): 671680. doi:10.1126/science.220.4598.671. JSTOR.*

[35] Dongarra, Jack J and Luszczek, Piotr and Petitet, Antoine, "The LINPACK Benchmark: past, present and future,"*Concurrency and Computation: Practice and Experience*, vol. 15, no. 9, pp. 803-820, 2003