Optimizing Job Reliability via Contention-Free, Distributed Scheduling of VM Checkpointing

Yu Xiang
George Washington University
Washington, DC, USA
xy336699@gwu.edu

Hang Liu
George Washington University
Washington, DC, USA
asherliu@gwmail.gwu.edu

Tian Lan
George Washington University
Washington, DC, USA
tlan@gwu.edu

Howie Huang
George Washington University
Washington, DC, USA
howie@gwu.edu

Suresh Subramaniam
George Washington University
Washington, DC, USA
suresh@gwu.edu

ABSTRACT

Checkpointing a virtual machine (VM) is a proven technique to improve the reliability in modern datacenters. Inspired by the CSMA protocol in wireless congestion control, we propose a novel framework for distributed and contention-free scheduling of VM checkpointing to offer reliability as a transparent, elastic service in datacenters. In this work, we quantify the reliability in closed form by studying system stationary behaviors, and maximize the job reliability through utility optimization. We implement a proof-of-concept prototype based on our design. Evaluation results show that the proposed checkpoint scheduling can significantly reduce the performance interference from checkpointing and improve reliability by as much as one order of magnitude over contention-oblivious scheme.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*checkpoint scheduling optimization*

Keywords

Data center, checkpoint, scheduling, reliability, CSMA-based, optimization

1. INTRODUCTION

In today's cloud, reliability is often provided as a fixed service parameter, e.g., all Amazon EC2 users are expected to receive 99.95% reliability [1]. However, users may find it either too inadequate or too expensive to fit their various reliability requirements, thus providing elastic reliability for the masses remains an elusive goal in cloud computing today. A number of models for calculating the optimal checkpoint schedule [2,3] has been proposed for reliability optimization of *a single job*, but these solutions fall short in optimizing checkpoints of *co-located jobs* that reside on the same physical server, where checkpoints from different jobs may introduce severe contention on shared resources.

In this work, we utilize the technique of saving local VM image before transferring to networked storage. The time to save local checkpoint images is determined largely by how I/O resources are shared, while the time to transfer locally saved images to networked storage relies on how network resources are shared. Note that each job may also consists of multiple VMs that reside on different physical servers. We use the term *serving host* to refer to a physical server that hosts at least one of the VMs belonging to the job. In a server that hosts hundreds of VMs from various jobs, chances are that VM checkpointing, if unmanaged and uncoordinated, would encounter severe network and I/O congestion, resulting in high VM checkpointing overhead and reliability loss.

The main contributions of this paper are three-fold:(i) We harness CSMA-based interference management policy to provide a distributed and contention-free checkpoint scheduling protocol (ii)Reliability received by each individual job is characterized in closed form. It enables a joint reliability optimization subject to flexible service-level agreements (SLAs) of all jobs (iii)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 sho-wn to significantly reduce checkpoint interference and improve reliability.

The paper is organized as follows. Section 2 illustrates the necessity for distributed, contention-free checkpoint scheduling, Section 3 introduces the protocol and analyzes joint reliability optimization. Section 4 show experiment and evaluation results, and Section 5 concludes.

2. MOTIVATION

In this paper, we address the problem of joint reliability maximization by developing a novel, distributed algorithm for scheduling job checkpoints in order to mitigate not only contentions among VM checkpoints but also interference between checkpoints and regular jobs. In the latter case, if a regular job is I/O or network intensive, the checkpoints on the same host would take longer to complete due to resource interference, necessitating lower checkpoint rates to avoid high overhead and inferior reliability. Now consider two extreme cases for multi-job checkpoint scheduling: parallel and pipeline scheduling, as illustrated in Figure 1, in which T_0 denotes the scheduling overhead, T_n represents the time to take a checkpoint and T_f is the time taken to transfer checkpoint image to remote servers. In the parallel mode, the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
DCC'14, August 18, 2014, Chicago, Illinois, USA.
Copyright 2014 ACM 978-1-4503-2992-7/14/08 ...$15.00.
http://dx.doi.org/10.1145/2627566.2627568.

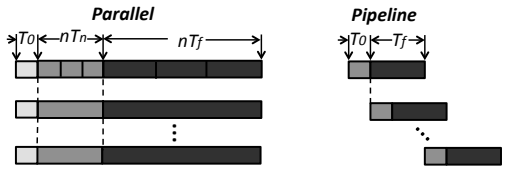


Figure 1: Fully coordinated checkpoint scheduling in a pipeline mode significantly reduces resource contention over parallel checkpoints.

checkpoints of all N jobs are done at the same time and the total I/O and network bandwidth are shared among them. This represents the case with high resource contention in checkpointing. On the other hand, the checkpoints of jobs can be taken one immediately after another in a pipelined fashion by overlapping the image-saving time of one job's checkpoint with the transfer time of another job. With such completely coordinated checkpoints, the contention between the checkpoints is reduced and the jobs can take full advantage of all I/O and network bandwidth resource available with minimal interference to others. Our experiment shown in Figure 3 proves that such pipeline checkpoint scheduling improves the reliability by nearly an order of magnitude for various VM sizes over parallel scheduling.

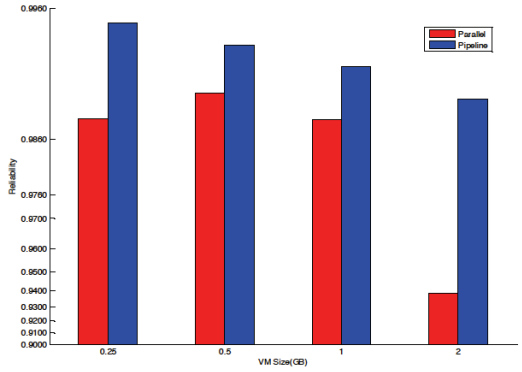


Figure 3: Fully coordinated pipeline checkpoint schedule significantly reduces contention and improves reliability over parallel checkpoint schedule. Reliability calculated with 8 failures/year.

To summarize, even though pipeline scheduling completely avoids checkpoint interference, such a centralized coordination and micro-management approach is prohibitive in large-scale datacenters. A practical checkpoint scheduling scheme should (i) allow a distributed implementation, (ii) schedule contention-free checkpoints for a large number of jobs that may have varying 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 to derive a distributed, contention-free checkpoint scheduling protocol with joint reliability optimization.

Reliability Model. As shown in Figure 2, a single job checkpoints all its VMs every T_i seconds. We have $T_f = 0$ (since no additional transfer time is needed) while T_n increases significantly because checkpointing to a networked storage takes much longer than checkpointing locally. In our framework, jobs can be restored from an available checkpoint and rolled back to the last saved state with recovery time T_r when a failure happens. We define reliability as the value of one minus the fraction of expected service down-

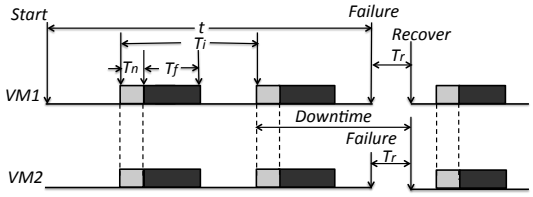


Figure 2: Multiple VMs belonging to the same job must be checkpointed simultaneously to avoid cascaded roll-backs. It increases the chance of checkpoint contention.

time. More precisely, let a failure occur at time t after the n-th checkpoint is fully completed. Then,

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

where nT_n is the total service downtime due to taking checkpoints, when a failure happens at time t, it needs to roll back to the end of last successful checkpoint, $t-(n-1)T_i-T_n-T_f$ is the lost service time due to roll-back, where $t-(n-1)T_i$ is the roll-back time to last checkpoint and $T_n + T_f$ is the duration of last checkpoint and is considered as up-time since the job would roll back to the end of last checkpoint. It is easy to see that there exists a trade-off between checkpoint interval T_i and reliability R. Since the reliability improves as T_i increases due to lower checkpoint frequency and overhead.but the expected roll-back time to the last checkpoint also increases as the checkpoint frequency decreases.

3. PROTOCOL AND RELIABILITY OPTIMIZATION

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., $\mathcal{H}_i \subseteq \mathcal{S}$. Since each job may consist of multiple VMs distributed to different physical machines, to guarantee consistency in our design, checkpoints are organized at the job level - to checkpoint a job, images of all its VMs must be created and saved to remote networked storage to avoid host failure. Our CSMA-Based 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. The implementation details will be described in Section 4. 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 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 [6]. In such an idealized CSMA model, if sensing time is negligible and back-off time follows a continuous

[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.

distribution, then the probability for two contending checkpoints to start at the same time is 0 [5]. Therefore, the CSMA-based protocol achieves contention-free, distributed scheduling of job checkpoints.

3.1 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. 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, for any time t, we define a system state as the set of jobs actively taking checkpoints at t, in each state, a set of non-conflicting jobs are scheduled. We assume that there exist $K \leq 2^N$ possible states, represented by $\mathcal{X}_k \subseteq \mathcal{N}$, for $k = 1, \ldots, K$. In state $\mathcal{X}_k$, if job i is not taking checkpoints and all of its conflicting jobs are not taking checkpoints, the state $\mathcal{X}_k$ can transit to state $\mathcal{X}_k \cup \{i\}$ with a rate λ_i (i.e., job i starts its checkpoint). Similarly, state $\mathcal{X}_k \cup \{i\}$ can transit to state $\mathcal{X}_k$ 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).

Our goal is to quantify job reliability using this 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 apply the uniformization technique to obtain a randomized Discrete Time Markov Chain (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:

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

where $P_{\mathcal{X}_k, \mathcal{X}_l}$ denote the transition probabilities from state $\mathcal{X}_k$ to state $\mathcal{X}_l$. Due to uniformization, we define $v_k = \sum_{l \neq k} v \cdot P_{\mathcal{X}_k, \mathcal{X}_l}$ to be the sum of transition probabilities out of state $\mathcal{X}_k$ so we have

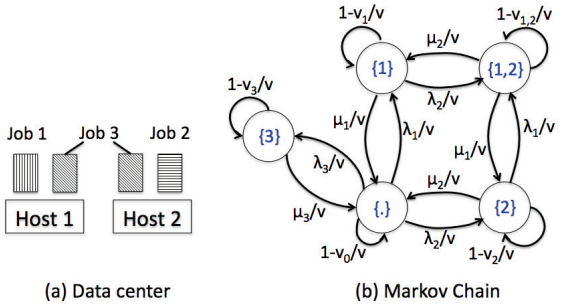
$$P_{\mathcal{X}_k, \mathcal{X}_k} = 1 - \frac{v_k}{v}. \qquad (3)$$

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. 4 (a) gives an example datacenter with 3 jobs and 2 hosts. If each host is able to checkpoint one 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 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.

The stationary behavior of the above DTMC model reveals the distributions of checkpoint overhead T_n, T_f, and checkpoint interval T_i. The stationary distribution is denoted by $\pi_1, \ldots, \pi_K$, satisfying

$$(\pi_1, \ldots, \pi_K) = (\pi_1, \ldots, \pi_K) \cdot P, \qquad (4)$$

where π_k is the stationary probability that the DTMC stays in state $\mathcal{X}_k$.



(a) Data center (b) Markov Chain

Figure 4: Example: 3 jobs and corresponding Markov Chain.

3.2 Reliability Analysis

Now that we have π_k of the Markov model to obtain the distribution of T_n, T_f, T_i, we further 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 [7, 8]. 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 $\mathcal{A}_i = \{\mathcal{X}_k : i \in \mathcal{X}_k\}$ 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 $\mathcal{A}_i$, i.e., the time to checkpoint job i's VMs. Similarly, checkpoint interval T_i is the first returning time of the CTMC to $\mathcal{A}_i$. 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.

We can quantify the reliability received by each job i in our contention-free, distributed checkpoint scheduling protocol as:

THEOREM 1. *For given rates $\lambda_1, \ldots, \lambda_K$, job i has Poisson failures with rate f_i, each job i in our protocol receives the following reliability R_i:*

$$R_i = 1 - f_i \tau_i^r - \tau_i^c \mu_i \pi_{\mathcal{A}_i} - \frac{f_i}{\mu_i}(\pi_{\mathcal{A}_i} + \frac{1}{\pi_{\mathcal{A}_i}}) \qquad (5)$$

where τ_i^c is the mean time to save a local checkpoint image, τ_i^r is mean repair time, and $\pi_{\mathcal{A}_i} = \sum_{k \in \mathcal{A}_i} \pi_k$ is the sum of all states in $\mathcal{A}_i$. The proof and derivation of this theorem can be found in our online technical report [11].

3.3 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 an arbitrary non-decrasing 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 [10] that maximizes total utility $\sum_i U_i(R_i)$, i.e.,

$$\begin{aligned} \max \quad & \sum_i U_i(R_i) \qquad (6) \\ \text{s.t.} \quad & R_i = 1 - f_i\tau_i^r - \tau_i^c\mu_i\pi_{\mathcal{A}_i} - \frac{f_i}{\mu_i}(\pi_{\mathcal{A}_i} + \frac{1}{\pi_{\mathcal{A}_i}}), \\ & \pi_{\mathcal{A}_i} = \frac{1}{C_\lambda} \cdot \sum_{\mathcal{X}_k \in \mathcal{A}_i} \prod_{j \in \mathcal{X}_k} \lambda_j \cdot \prod_{l \notin \mathcal{X}_k} \mu_l, \\ \text{var.} \quad & \lambda_1, \ldots, \lambda_K \end{aligned}$$

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

The reliability optimization is computed by maximizing an aggregate utility $\sum_i U_i(R_i)$ over all feasible sensing rates $\lambda_1, \ldots, \lambda_K$. The optimal solutions are obtained through either local search heuristics or other sufficient conditions, then 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 reliability analysis in Theorem 1, we implement a prototype of the contention-free, distributed checkpoint scheduling protocol. The implementation and evaluation details are provided later in Section 4. 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 $i = 1, \ldots, 24$(experiment details are provided later in Section 4), and reliability in the experiment is obtained according to (1) using the parameters from the benchmark. So all jobs in the experiment receive equal reliability value. For a sensing rate of $\lambda_i = 1/(2.5 \text{ days})$ 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 5 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 datacenter 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 $f_1 = 10 \cdot f_0 = 6.43 \times 10^{-7}$ and small jobs $f_2 = 2 \cdot f_0 = 1.29 \times 10^{-7}$. 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 [13] to find the optimal sensing rates λ_1, λ_2 that maximize a utility $U_i(R_i) = 2R_1 + R_2$. As shown in Figure 6, 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 $R_2 = 0.90$ because the weight of large jobs is twice as that of small jobs in the optimization objective $2R_1 + R_2$.

4. IMPLEMENTATION AND EVALUAT-IONS

We have implemented a prototype of the contention-free checkpoint scheduling based on Linux and Xen. Our scheduling strategy is achieved with a locally managed list of the checkpointing status for all the VMs. Each VM will check the co-located VM's status through this list before checkpointing in order to avoid the contention. When no others are checkpointing, the VMs belonging to the same job will update their checkpoint status to *checkpointing* and start the checkpointing process. Once the checkpointing is done, the VMs will update the status to *non-checkpointing*.

For testing, we use a local cluster where each node has an Intel Atom CPU D525 processor, 4GB DRAM, 7200 RPM 1TB hard drive, and 1Gb/s network interface. Note that I/O and network bandwidth rather than CPU and memory are the major limiting factors for our tests. To simulate the workload, each VM runs a CPU-intensive benchmark [12] with 1 VCPU, 512MB or 1GB DRAM, and 10GB VDisk. The host OS is Linux 2.6.32 and Xen 4.0. Each failure is simulated by manually killing a VM. If not specified, the failure rate is eight times per year, and each reliability result is the average of three runs. We compare our contention-free scheduling with contention-oblivious scheduling where each job independently performs checkpointing at the predefined intervals.

Figure 7 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 reliability for both contention-oblivious and contention-free scheduling is 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 8. Overall, contention-free scheduling can achieve a reliability of two nines ($> 99\%$), compared 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 and 256 VMs. In this experiment, we intentionally intensify the job checkpointing rate in our cluster. As shown in Figure 9, 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 10. 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 year. One can see that our contention-free checkpointing can achieve a reduction in downtime of upto 18.3% compared to contention-oblivious scheduling.

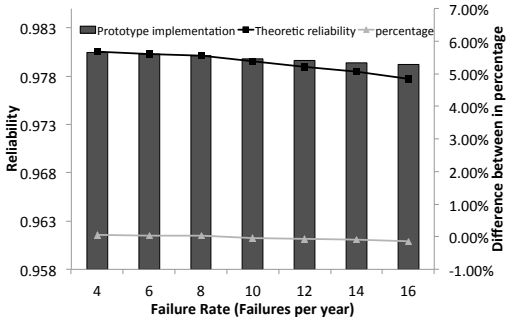


Figure 5: Comparison of the reliability values from our theoretical analysis with a prototype experiment using 24 VMs in Xen.

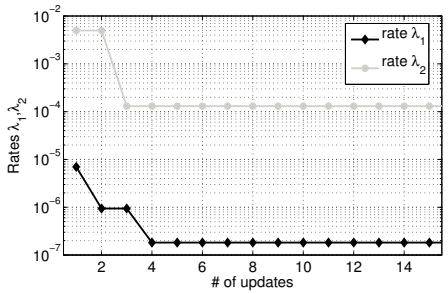


Figure 6: Convergence of sensing rates λ_1, λ_2 when Hill Climbing local search is employed to solve the optimization.

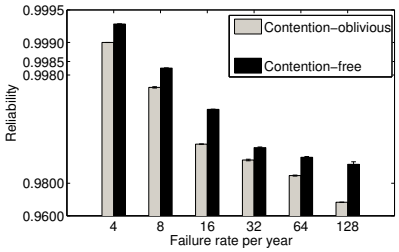


Figure 7: Reliability for different failure rates

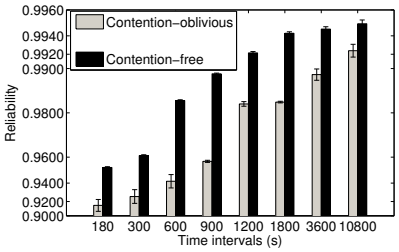


Figure 8: Different checkpoint time intervals

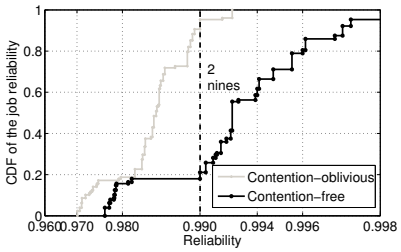


Figure 9: Checkpointing 128 jobs

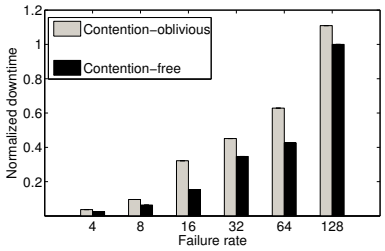


Figure 10: Normalized downtime for different annual failure rates.

5. CONCLUSIONS

Inspired by the CSMA protocol, we propose a new protocol for distributed and contention-free checkpoint scheduling which also provides elastic reliability service to meet disparate user-requirements in large-scale datacenters. 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 checkpoint scheduling are demonstrated via experiments in realistic settings.

6. REFERENCES

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

[2] 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.

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

[4] A. Kangarlou et al, "In-network Live Snapshot Service for Recovering Virtual Infrastructure," *IEEE Network*, vol. 25, no. 14, pp. 12-19, 2011.

[5] 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.

[6] S.C. Liew et al, "Back-of-the-Envelope Computation of Throughput Distributions in CSMA Wireless Networks," *submitted for publication http://arxiv.org//pdf/0712.1854*.

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

[8] 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*

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

[10] M. Chiang et al, "Layering as Optimization Decomposition: A Mathematical Theory of Network Architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255-312, 2007.

[11] Y. Xiang et al, "Optimizing Job Reliability Through Contention-Free, Distributed Checkpoint Scheduling," *Online technical report available at www.seas.gwu.edu/ tlan /papers/ICAC.pdf*, 2013.

[12] 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.

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