Vol 5 Issue 5 Nov 2015 ISSN No :2231-5063

International Multidisciplinary Research Journal

Golden Research Thoughts

Welcome to GRT

RNI MAHMUL/2011/38595 **ISSN No.2231-5063**

Golden Research Thoughts Journal is a multidisciplinary research journal, published monthly in English, Hindi & Marathi Language. All research papers submitted to the journal will be double - blind peer reviewed referred by members of the editorial board.Readers will include investigator in universities, research institutes government and industry with research interest in the general subjects.

Address:-Ashok Yakkaldevi 258/34, Raviwar Peth, Solapur - 413 005 Maharashtra, India
Cell : 9595 359 435, Ph No: 02172372010 Email: ayisrj@yahoo.in Website: www.aygrt.isrj.org

Golden Research Thoughts
ISSN 2231-5063

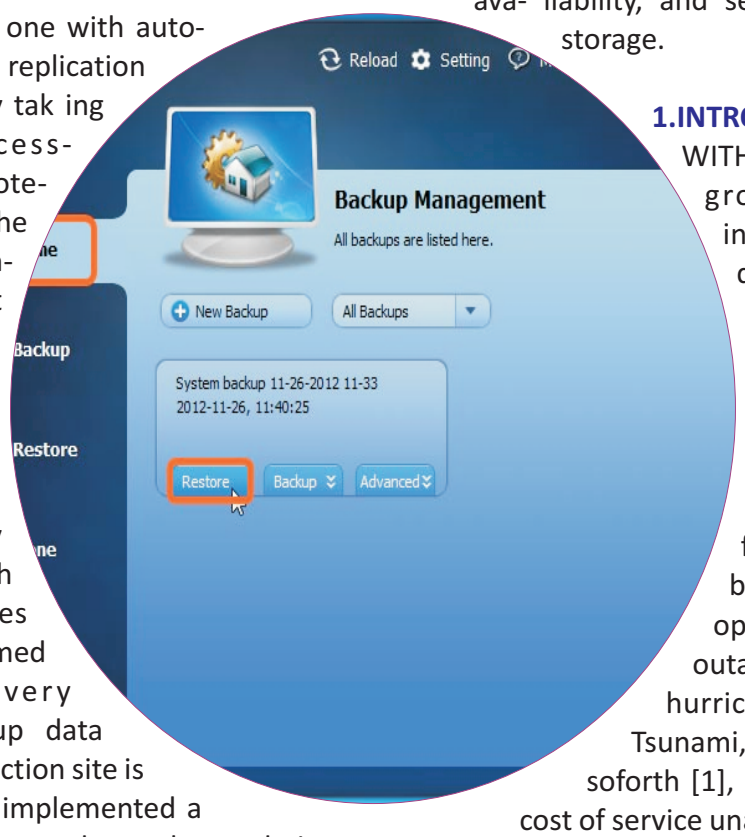
Impact Factor : 3.4052(UIF)
Volume - 5 | Issue - 5 | Nov - 2015

A SURVEY OF BACKUP RECOVERY SYSTEM FOR INSTANT RESTORATION OF DATA SERVICES

[1]Nitin D. Magdum and [2]Manik A Raichurakar
1Student, Department of Computer Engineering, JSPM Narhe Technical Campus, Pune, Maharastra, India Savitribai Phule Pune University, Pune.
2Professor, Department of Computer Engineering, JSPM Narhe Technical Campus, Pune, Maharastra, India. Savitribai Phule Pune University, Pune.

ABSTRACT:
W e propose Birds: a bare-metal recovery system for instant restoration of data services, focusing on a general-purpose automatic backup-and-recovery approach to protect data and resume data services from scratch instantly after disasters. We design BIRDS to possess two appealing features: full automation in the backup-and-recovery process, and instant data service resum- ption after disasters. BIRDS achieves the former one with automatic whole system replication and resto- ration, by tak ing the backup process-outside of the protected system with the help of a novel non-intrusive light-weight physical to virtual conversion method. The latter oneis enabled by a novel pipelined parallel r e c o v e r y mechanism, which allows data services being instantly resumed while data recovery between the backup data center and the production site is still in progress. We implemented a BIRDS prototype and evaluated it usingstandard benchmarks. We show that BIRDS outperforms existing disaster recovery techniques by the means of recovery efficiencywhile introducing relatively small runtime overhead. Furthermore, BIRDS can be directly applied to any existing system in a plug-and-protect fashion without requiring re-installation or any modification of the existing system.

KEY WORDS: Backup/ recove- ry, rel-i ab- ility, ava- ilability, and se-r viceability, mass storage.



1.INTRODUCTION:

WITH the expon- ential growth of Internet informati- onservices, data has become the critical asset thats hould be kept highly a v a i l a b l e a n d reliable. However, failures do occur such as hardware f a u l t s , s o f t w a r e bugs, virus attacks, operator errors, power outages, building on fire, hurricane, earth quake, Tsunami, terrorist attacks and soforth [1], [2], [3], [4], [5]. The cost of service unavailabilitycaused by such failures can be as high as $1 million perhour [4], [6], [7], which is disastrous to businesses. In orderto protect data from possible disruptions

and to recoverdata services in case of failures, disaster recovery (DR) technologies are developed.

Although DR technologies provide substantial potential benefits and many approaches have been proposed, protecting data and resuming data services quickly toensure business continuity after disasters have been difficult in practice. There are four important and hard goalsthat need to be met. First, smaller recovery time objective(RTO) is desired, which indicates an instant serviceresumption to minimize service down time. Second,smaller recovery point objective (RPO) is expected, whichmeans shrinking time window between two successivebackup operations to minimize possible data loss due tonot timely data backup. Third, given the rapid growth ofreal world data services, the backup-and-recovery process should be general-purpose and application independent. Fourth, backup-and-recovery operations shouldonly introducelow overhead that is negligible for normalrunning of data services, and the deployment of DR systems should be non-intrusive to normal data services.

In this paper, we focus on the “bare-metal” DR problem,which aims to restore data services from scratch after sitelevel failures occur and all processes and storage systemsresiding on a site are corrupted. In such conditions, small RTOis especially difficult to be achieved as the typicalrecovery time for data services when facing site-level failures is too long to tolerate. Hence, our proposed DR approach focuses on minimizingRTOwhile keeping itself general-purposeand withlow overhead. As recent techniqueadvances in terms of shorteningRPO[13], [17], [29] can beapplied to our approach with no technical conflicts, we do not explicitly address the problem of minimizingRPOin this paper.

Current DR approaches can be grouped into two categories according to their means of recovery:FailoverandFailback. However, in terms of achieving small RTO, they suffer from their own limitations when facing bare-metal scenarios. Failover approaches like storage mirror and Virtual Machine (VM) based high availability ensuring methods[8], [9], [10], [11], [29] use a backup site as a hot standby.They can reduce the disaster recovery time to near-zero byswitching the service to the backup site when the production site corrupts. However, these methods are expensive asthe remote backup site obviously needs to be constructedwith task processing power, data storage capacity, and network bandwidth compatible with original production site at all time.

As an alternative,Failbackmethods used by common backup-and-recovery systems [12], [13], [14], [15], [16] are much less expensive which is focused to backup the persistent storage data from the production site to the backup site, and restore services in the production site by using remote backups. Among them, continuous data protection (CDP) is the main failback method that can be used to shorten the recovery time after disasters. Theoretically, a CDPsystem performs fast recoveries by minimizing the data set to be restored. It recovers only the corrupted data blocks. Apparently, fast recovery that is promised byCDP fails when site-level failures ruin the whole service site. In such conditions, they suffer from unavoidable manual operations like system reconfiguration that prohibit automatic recovery, and the service down time is long as they recover the entire persistent storage data before resuming data service. Virtual Machine basedfailback methods [17], [18], [19] provide automatic backup-and-recovery of running states and persistent storage data. However, they cannot be applied to general production systems without VM support. Data services should be reinstalled in a VM-ready environment which is too intrusive to existing services, and most such methods also fail to accomplish the goal of instant recovery after corruptions [17], [19].

We present BIRDS, a bare-metal recovery system for instant restoration of data services in the paper. BIR is a general-purpose automatic failback approach to protect data and resume data services instantly after disasters. BIRDS targets a bare-metal recovery scenario, in which recovery process restores a computer system from scratch, i.e., without any requirements as to previously installed softwares or OS.

We have two technical contributions. First, BIRDS automates the backup-and-recovery process by isolating the operating system where backup processes resides (backup OS) from the protected operating system where data services reside (production OS). For the implementation, it uses a novel light-weight non-intrusive plug-and-protect method seamlessly converts the protectedproduction OSinto a slave OS of thebackup OS. The whole backup-and-recovery process of BIRDS is fully automatic which eliminates the time needed for human interventions with the help of another bootable disk drive.

Second, for instant service recovery, BIRDS introduces a novel recovery policy named as parallel recovery which achieves smallRTOby scheduling the data retrieval process according to their emergencies. Data service is immediately restarted as soon as enough running state information is restored at the production site. The resumed service accesses data without notable additional latencies by pipelining the data retrieval process combined with on-demand fetch and speculative data prefetch.Parallel recoverymechanism tackles three main problems in BIRDS: storage page miss identification, data retrieval order determination and IO request processing.

We implemented a BIRDS prototype in Linux, which can be easily plugged into any server to be protected. After disasters, BIRDS can restore the protected system to a different machine from bare-metal, and resume its services almost immediately. The restored system can be loaded onto a new sever with hardware configuration not identical to the original server, as long as its CPU is of thesamearchitectureastheoriginalone,anditsmemory space is equal to or larger than the original one. We evaluate its recovery time, service performance, and runtimeoverheads by using two standard benchmarks (TPC-C for databases and SPECWeb for web services). The experimental results show that BIRDS can automatically resume data services within a few seconds after a failure event asopposed to thousands of seconds down time with existing DR techniques while the overhead is negligible.

The paper is organized as follows. Section 2 presents thegoals and key ideas of BIRDS. Section 3 introduces thedesign and implementation of BIRDS. Section 4 describes our experimental settings, workload characteristics, and experimental results. We summarize related work in Section 5 and conclude the paper in Section 6.

2.OVERVIEW

2.1 FAILURE DEFINITION

Assume that a production site hosts adata service by maintaining some processes to serve end users that exchange data to and from local persistent storage devices on theproduction site. The failure that we aim to recovery from issite-level failure resulted by serious accidents or natural disasters. Since such failures can be catastrophic, we build our backup-and-recovery system based on the following assumptions.

Remote backup-and-recovery.As the whole site can becorrupted, all the data and system states are routinely replicated to a remote backup data center. After disasters corrupt the production site, a recovery program is started tofetch back data and states from the remote backup datacenter and rebuild the data service. It is time-consumingto fetch data back from the remote backup data center,hence how to minimize the timefordataretrievalbefore rebuilding the data service is a key challenge that we need to tackle.:

Bare-metal recovery.As the failure can cause data loss or system damages on any server on the production site, no prerequisites as to previously installed operating systems orsoftwares are specified for starting the recovery process. Inother words, a recovery program can restore a production system from scratch.

2.2 GOALS

The goal of disaster recovery is to prevent data loss and torestore data services available after corruptions. BIRDSaims to restore data services from bare-metal instantly after disasters by designing efficient backup-and-recovery policies to meet the following goals:

Automatic and Instant Service Rebuilding.The backup-andrecovery process should be automated to eliminate human interventions that prolong the recovery process and potentially bring manual risks. Furthermore, service downtime should be minimized. Ideally, the data service resumes almost immediately once the hardware has been repaired.

Generality.The backup-and-recovery process needs to be application independent to ensure its generality, as real world data services emerge and change rapidly. That means YU ET AL.: BIRDS: A BARE METAL RECOVERY SYSTEM FOR INSTANT RESTORATION OF DATA SERVICES 1393 it can be applied to any data service without any modifications on neither itself nor the data service.

Low Overhead.The deployment of DR methods should be non-intrusive to production sites and the runtime overhead introduced by backup-and-recovery operations should be small enough for practical data services.

2.3 ELIMINATE HUMAN INTERVENTIONS: TAKE THE BACKUP AGENT OUT

The first insight of this paper is that the prerequisite for high efficient bare-metal recovery is to take the backup agent out of the OS where protected data services reside. Because it is a practical way to eliminate human interventions during the backup-and-recovery process. Human interventions slow down the recovery process, bring manual risks, and prohibit automatic and instant recovery after disasters. In order torestoredataservicesaftera failure event, typical bare-metal recovery process involves five main steps after the hardware is repaired: (1) boot from external drive and start the recovery process, (2) restore data from the backup data center to the production site, (3) reboot the production site, (4) resume data services on the production site, (5) restart the backup agent for the next failure. From them, we identify three kinds of human interventions that are unavoidable when using traditional data-oriented DR solutions for baremetal recovery: system reboot, application restart, and system/applicationreconfiguration.

These manual steps are essentially to restore data services to their previous OS running context and normal service running states. Hence, the preconditions for eliminating these manual steps during recovery are to replicate the OS running context together with normal service running states to another place and to restore them automatically. The failover DR solutions maintain a consistent copy of current OSrunning states and service running states with full redundant system and infrastructures. As a result, they can switch to the secondary site directly without human interventions. It is more difficult forfailback DR solutions though, as they now need to backup not only persistent storage states and service running states, but also volatile OS running states to a network volume, and restore them when recovering from disasters. Traditional DR solutions fail to do so, because thebackup agent(who is in charge of states replication) is running in the same OS where the protected data services reside and thus not capable of replicating the entire OS running states.

More precisely, we name the OS where the backup agent resides asbackup OS, and the OS where protected data services reside asproduction OS.IntraditionalDR solutions,backup OSis the production OS.Inorderforthe backup agent to be able to backup the entire running states of the production OS, we need to take the backup agent out of the box composed of the production OS.
Towards this end, we use a non-intrusive method to convert the existingproduction OS into a slave OS of the backup OS, and then replicate both persistent storage states together with volatile OS running

states by the use of a virtualization based OS checkpoint technique. With that in hand, even in a bare-metal scenario, data services can be backed up and recovered automatically.

2.4 ENABLE INSTANT RECOVERY: PARALLEL THE RECOVERY PROCESS

The main idea for instant recovery comes from an observation that the recovery process in traditional data-oriented DR solutions can be divided into two stages: data retrieval and service recovery, which can be clearly identified in Keeton's recovery graph [7]. More importantly, service recovery cannot be started until data retrieval is finished. If recovering data services waits for data to be restored completely before taking over the incoming requests, the service down time will be long inevitably. The implicit assumption for such recovery strategy is that data priorities are the same for all data. Thus we need to recover all data completely before service resume.

To enable instant recovery, we propose a novel recovery strategy, parallel recovery, which performs data retrieval and service rebuilt in parallel and allows data restoration ordered by priorities. As soon as the minimum data required for launching service threads is retrieved from the backup data center, service rebuilt starts immediately and progresses with data restoration in parallel from then on

With the strategy ofparallel recovery, data are queued andfetched back according to their emergencies. The order of data retrieval can be determined by the actual requests sent by service threads (i.e.,on demandpolicy) or by employing predictions based on data usage information to further reduce wait time (i.e.,pipelinedpolicy). Ideally, if we can predict the order of data needed correctly and retrieve them in time, the delay caused by missed data can be totally avoided.

Compared with traditional recovery solutions,parallel recoveryshortens the service down time, but causes possible service degradation for the residual dependency [20] between backup and production data center.

Parallel recoveryis different from VM based system migration too. It can be viewed as a live migration method that migrates already suspended whole system snapshots stored in the backup site back to the production site. It differs from pre-copy VM migration, such as VMware ESX [11], in the sense that pre-copy VM migration initially copies the entire disk on the source node to the destination node without anydata priority considerations, and iteratively propagates data changes being modified on the source to the destination later.

On the other hand, current post-copy VM migration [20] is more similar to our approach but only focuses on running states migration. Post-copy [20] copies minimal processor states to the destination node, resumes the VM, and begins to fetch memory pages on demand over the network. Snowflock [36] uses a similar method like post-copy. However, it does not handle persistent storage replication.

Whereas,parallel recoveryis a whole system migration method focused on challenges generated by storage migration issues including identifying storage page misses, determining the order of data retrieval and processing IO requests. VMLaunchPad and VM Profiler in VMFlock [37] share more similar ideas with parallel recovery. However, VMFlock deploys a static profiling based data retrieval policy to fetch back data blocks need mostly for system 1394 IEEE TRANSACTIONS ON COMPUTERS, VOL. 63, NO. 6, JUNE 2014 rebooting, and retrieves missing blocks on demand, while Parallel recoverydepends on realtime dynamic prediction based data retrieval policies.

3.DESIGN ANDIMPLEMENTATION

We first describe our design decisions on encapsulating the production OS because it is essential to understand the layout of the overall architecture of Birds.

3.1 ENCAPSULATE THE PRODUCTION OS

The backup agent of BIRDS is in charge of monitoring the production OSfor tracing the changes about both persistent storage states and volatile running states, and propagate the changes to the backup site. There are three main ways to trace and record the running states: process checkpoint [21], [22], container based virtualization [23], and virtual machines [24], [25]. To take the backup agent out of theproduction OSwhile keep tracing, it is feasible to use either container based virtualization or the virtual machines. A trivial design to implement this is to boot from the backup OS, and reinstall the production OSas a slave OS. This is really intrusive and inoperable for most already-running data service systems.

As an alternative, BIRDS introduces a non-intrusive plug-and-protect approach. The idea is to add another disk drive (may be attached as an external USB drive or embedded as another new SCSI/SATA drive) to the existing system, and we name it “backup drive”. The drive is installed with a bootable OS with virtualization support. This bootable OS is thebackup OSand the backup agent resides in it. When rebooted from it, a physical to virtual OS transformation process will be started, which converts the production OSon the to-be-protected server to a virtualized subsystem of the OS resided in thebackup driveof the same physical host. As a result, theproduction OSbecomes a slave OS of the backup OS, and is totally controlled by it. The system will be booted into thebackup OSat all times after the installation, and theproduction OSwill be booted from thebackup OSfrom then on.

We choose to use container based virtualization instead of VMs to encapsulate theproduction OSfor two reasons. The first is that our purpose here is data protection and recovery rather than resource isolation which is the focus of virtual machines, thus smaller runtime overhead after encapsulation is important, which is the advantage of container based virtualization techniques. For example, in a VMware official performance report for Oracle database [38], the overall performance overhead of VMware VM is about 15 percent, which is nonnegligible. The second is that the physical to virtual transformation process of container based virtualization techniques is more efficient, which leads to faster and easier installation of the backup-andrecovery system. On contrary, virtual machines require a VMware P2Vlike method to decouple and migrate a physical server’s OS, applications, and data from a physical server to a virtual machine guest hosted on a virtualized platform. During the process, a full disk copy and complicated format transformation are involved because virtual machines usually use their own disk image format.

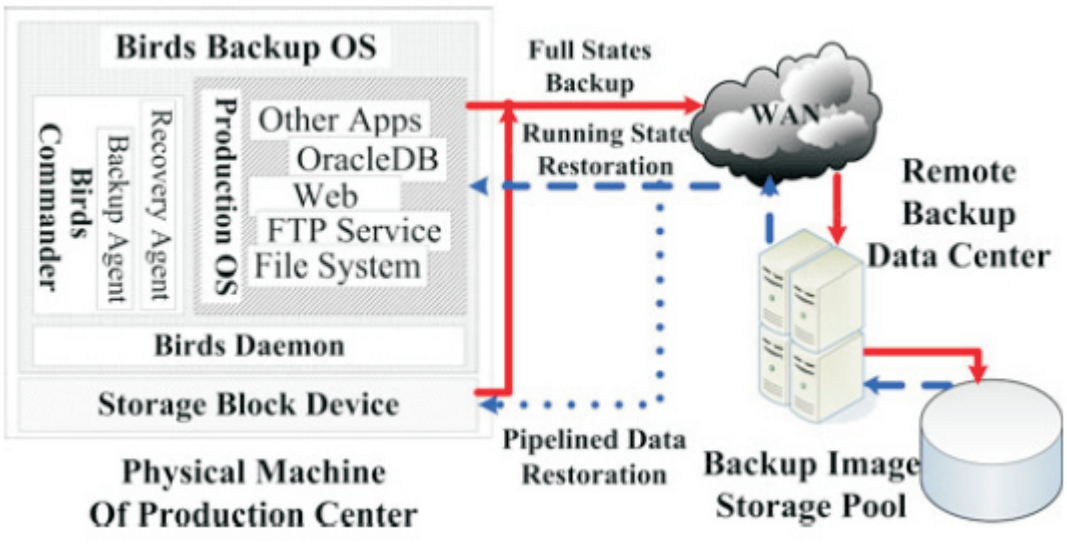


Fig. 1. BIRDS architecture. Solid lines indicate the backup data flow while dashed lines mean recovery data flow. Volatile service running states are backed up and restored separately with the persistent storage d.ata.

The physical to container conversion in BIRDS is performed as follows. First, we need to construct the “root” of theproduction OS. To achieve this, all files in the originalproduction OSshould be “copied” to a specific directory of the container. Our transformation happens in the same physical machine, which enables us to use a data mapping method without physically copying data. Essentially, our encapsulation tool running in thebackup OSidentifies the main system partition of theproduction OSand mounts it as a directory of thebackup OS.

Second, we need to construct a new container which represents the production OS. In BIRDS, we add an “empty” container pre-positioned in the bootablebackup OSby adding a configuration file for the new empty container. Then the container’s configuration file is modified, so that the container’s main partition is pointing to the root directory prepared previously. Other system configurations of thephysical machine, including memory, disk, and network settings, are also collected from theproduction OS, and are written into the container’s configuration file in thebackup OStoo. With this process, the container’s configuration file in the backup OShas collected enough information and is ready to use.

Thirdly, after rebooting the physical machine from the backup drive, thebackup OSstarts the running of theproduction OS as its slave by using virtualization management tools. Then, a “chroot” operation is done for the production OSto make it running correctly. Note that having the production OSand thebackup OSof different Linux versions may cause compatibility issues. All applications in theproduction OSare programmed and complied in theproduction OS, whereas the protected server is running the kernel of the backup OS. As a result, system calls made by applications in the production Osmay not be executed correctly, if the two Linux versions have different interpretations for them. To solve this problem, we could make Birds support each Linux version, and let the user choose the one that is consistent with theproduction OS.

3.2 BIRDS ARCHITECTURE FOR BACKUP-AND-RECOVERY

Fig. 1 shows the overall structure for the backup-and-recovery operations of BIRDS. The protected production OS is sealed in a container controlled by the backup OSusing an OS virtualization method. A BIRDS Commander which composed of the backup agent and recovery agent is designed to be run in the application layer of the backup OS. It issues commands for backup or recovery when needed. Commands are handled by a BIRDS Daemonwhich resides in the kernel of the backup OS. The BIRDS Daemon monitors the running of the production OS, and backups or recovers the checkpoints of the production OS's running states together with the persistent storage states to and from the backup center upon the requests of theBIRDS Commander.

For initialization, a full copy of the persistent storage and the initial running states are replicated to the remote backup center. After that, any storage data changes are caught by theBIRDS Daemonand replicated incrementally.

For backup, check points of the protected production OSare stored in a file, together with incremental snapshots of the container's persistent storage stored in the form of logical volumes (LV). The file which contains the incremental replication of storage data and a logically full running states copy are sent to the remote backup data center periodically. We have not implemented the incremental replication mechanism for the running states yet, because the running states need to be recovered is much less than the persistent storage data in bare-metal recovery after site-level failures, and can be further compacted to much smaller sizesdue to redundancies naturally existe damongthem.Incrementalreplicationofthe running states can also be adopted in BIRDS to further reduce overhead of the backup process and shortenRPO as in virtual machine time traveling system [17] or high availability oriented system like Remus [29].

For recovery, after new hardware is prepared and ready to use, BIRDS can immediately start the recovery process in two stages. First, the backup OSis booted using the hardware and the prerecorded running states are transferred back from the remote backup center. Theproduction Osis then resumed using such states. In this stage, no persistent storage data is restored yet. Second, the BIRDS Daemon monitors the running of the resumed service which requests to access data in the local storage device. Data misses will happen soon because no data is really restored yet. The BIRDS Daemon fetches the data back from the remote ackup center according to data priorities determined by factors like temporal and spatial localities as described in Section 3.4.

The pipelined parallel processing of resumed applications and data recovery in BIRDS is made possible by three key designs inBIRDS Daemon. (1) Stop/resume the execution of related processes in the container when a data block is detected to be unrecovered. (2) Construct a map, called Storage Map (SMAP), from the container's storage space to the remote data backup center, so we can retrieve data for the next execution slice while the container is running. (3) Probe what data is in use right now and predict what is needed in the near future by intercepting and analyzing the container's I/O requests, so that we can take advantage of our caching and prefetching policy and dynamically adjust data retrieval order.

In summary, the framework of the BIRDS Daemonis illustrated in Fig. 2, with following key components. Details about the components will be introduced in the following sections.

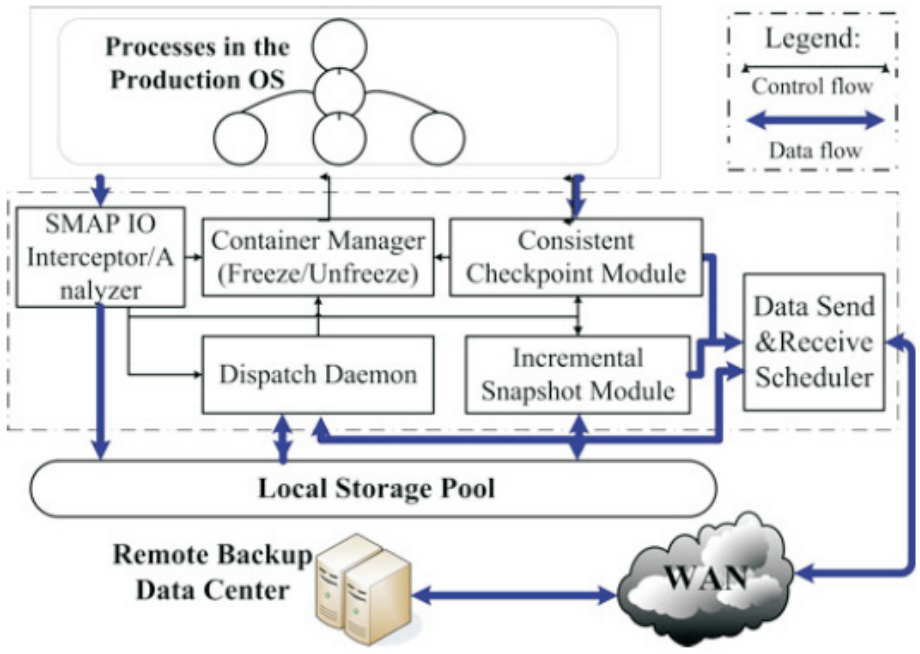


Fig. 2. The main components of BIRDS Daemon.

- Consistent Backup Module.This module is used to create checkpoints and to manage consistent backups including both Point in Time (PiT) process state checkpoints and storage space snapshots of the container. These consistent backups will be replicated to the remote backup data center for future recovery.
- Incremental Snapshot Module. This module is used to trace write operations, keep track of data changes, redirect I/O transparently, and create/manage storage snapshots.
- SMAP I/O Interceptor and Analyzer. This module traces I/O operations of running processes. It changes the data retrieval order dynamically according to gathered information from tracing and predefined policy.
- Dispatch Daemon.This module schedules overall data retrieval processes for parallel recovery. It receivesanalyzer's directive, sends data request to scheduler in background, and unfreezes the process when data is ready. It maintains logs of recovery status.
- Container Manager.It is used to freeze processes in
- the container both for checkpoint creation and parallel recovery. It also guarantees that, after unfreezing, the processes execute just from the point where it was frozen.
- Data Send & Receive Scheduler.Thismoduleisin charge of establishing connection with remote backup center, sending backup copies or data recovery requests, and more importantly, managing and scheduling all these requests fairly in an efficient way.

3.3 REPLICATING STATE CHANGES

In order to create application consistent backup points, BIRDS takes running state checkpoints and persistent storage snapshots simultaneously. The checkpoints and correlated snapshots are then transferred to a remote backup site with a unique time stamp for future recovery usages.

Before running state checkpoints can be taken, the container whereproduction OSand data

services reside in needs to be frozen first. Differed from VM based freezing methods which can suspend the execution of slave OS directly and

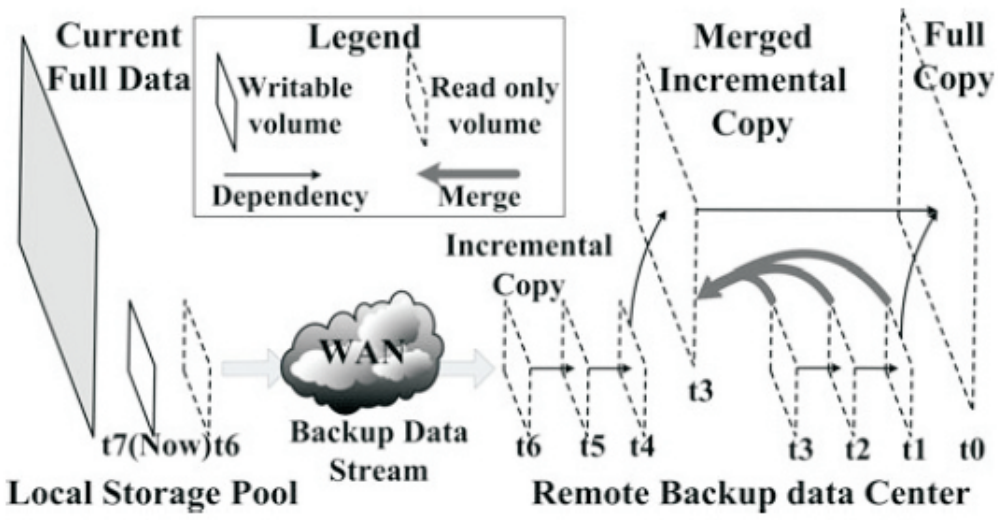


Fig. 3. Storage snapshot.

instantly, freezing the container in BIRDS is much more difficult. Each process in the container should be in a static state before we can dump its virtual memory area into checkpoint files.

In BIRDS, freezing the container is accomplished by the backup agent which initiates the freezing procedure, and the Container Manager freezing the processes in the container iteratively. When the backup agent initiates one round of checkpoint, it sends a freezing request to the Container Manager. The manager then iterates through all processes in the container and send a freezing signal to each process. These processes detect and handle the signals after returning from kernel to user space. They set their states to frozen and yield execution privilege, implying that these processes will never be candidates for process scheduling until being woken up.

Furthermore, after sending freezing signals, the Container Manager loops to check each process's state until the whole container is frozen. It is worth noting that some special processes need to be treated differently. For example, we must skip all thezombie, stopped, and ptracedprocesses in the container. In addition, the wait relationship among processes reveals a partial ordering that need to be kept after checkpointing, i.e., a processAdepends onBif AwaitsB for some operations. Towards this end, we make sure that processes are frozen in this order. After the entire container is frozen, full system states are replicated to checkpoint files. As soon as checkpointing is over, all the processes in the container are woken up in the reverse order of the recorded partial ordering.

The container's persistent storage state is replicated using the well-known incremental snapshot mechanism [8]. We construct a virtual storage pool to accommodate heterogeneous devices. Snapshot data is stored in units of logical volume, as illustrated in Fig. 3. Each storage snapshot is also treated as a LV, which is allocated dynamically from the pool and can be made up of several physical partitions. The data mapping and redirection details are implemented and masked by the kernel driver of incremental snapshot module.

Both the production site and remote backup site implement snapshot mechanism. In the production site, an incremental snapshot module is used to create and manage snapshot logical volumes. LVs are essentially buffers for keeping track of all incremental data changes. As shown in Fig. 3,

at current time,t7, a new snapshot LVis created for tracing all data changed sincet7. At the same time, the older LVcreated at t6 becomes un-writable and it buffers all changed data betweent6 andt7. They will be transferred asynchronously to remote backup site. In the remote backup site, all snapshotLVs at different time points are organized and merged periodically. Clearly,LVat timeti depends on LVs at timet0 to ti 1. In order to avoid searching data in multipleLVs, a sequence ofLVs can be merged periodically as shown in Fig. 3.

3.4 PARALLEL RECOVERY MECHANISM

With the restoration of running states before failures, we resume system running context even before retrieving storage data. The time cost of such process is relatively small due to much smaller memory checkpoint size compared with the huge size of persistent storage snapshot. After that, data retrieval and service rebuilding are progressed in parallel.

Identifying Page Misses. Since service rebuilding starts before the entire persistent storage data is restored, we need a mechanism to record and identify which pages are missing. BIRDS first uses a storage map mechanism to establish a mapping between container's local storage and remote backup, so that every local data address can be translated to a remote data address. BIRDS then uses arecovery bitmap, where each bit indicates whether or not a local physical page is restored. After service rebuilding starts,SMAP I/O Interceptor, which is implemented in the generic block layer, intercepts all I/O requests and detect data misses based on the information in the recovery bitmap. If a page miss is detected, which means the local physical page has not restored from the backup center,SMAP I/O Interceptornotifies a page miss event toSMAP I/O Analyzerand the process which wait for the missed data will be blocked until the data has been restored. Note that data write operations are not blocked when data miss happens. Instead, write operations are submitted to low level drivers instantly as if there is no data miss.

Therecovery bitmapis maintained by theDispatch Daemon which implements data retrieving from the remote site directed by theSMAP I/O Analyzer. It needs to be implemented with special care as well, as it could take up huge memory resource when protecting large-scale persistent storage. For example, assume the size of a physical page is 2 12 bytes, which is the basic unit for memory allocation in the OS kernel. With each bit representing the recovery state of a page of the persistent storage data, 1 TB storage data needs 32 MB kernel memory totally. We adopt two mechanisms to store and search therecovery bitmapefficiently with low memory consumption. First, if a bitmap block contains continuous '1's or '0' s, it can be compressed to save memory. Because most of the data are fetched sequentially, such compression happens frequently. Second, instead of allocating memory for the entire recovery bitmapin the beginning, bitmap blocks are allocated on demand, and we use a red black tree to organize and search these bitmap blocks. In this way, the huge memory consumption is amortized during the recovery process.

As shown in Fig. 4, the data structure of a tree node, which covers several contiguous bitmap blocks, include

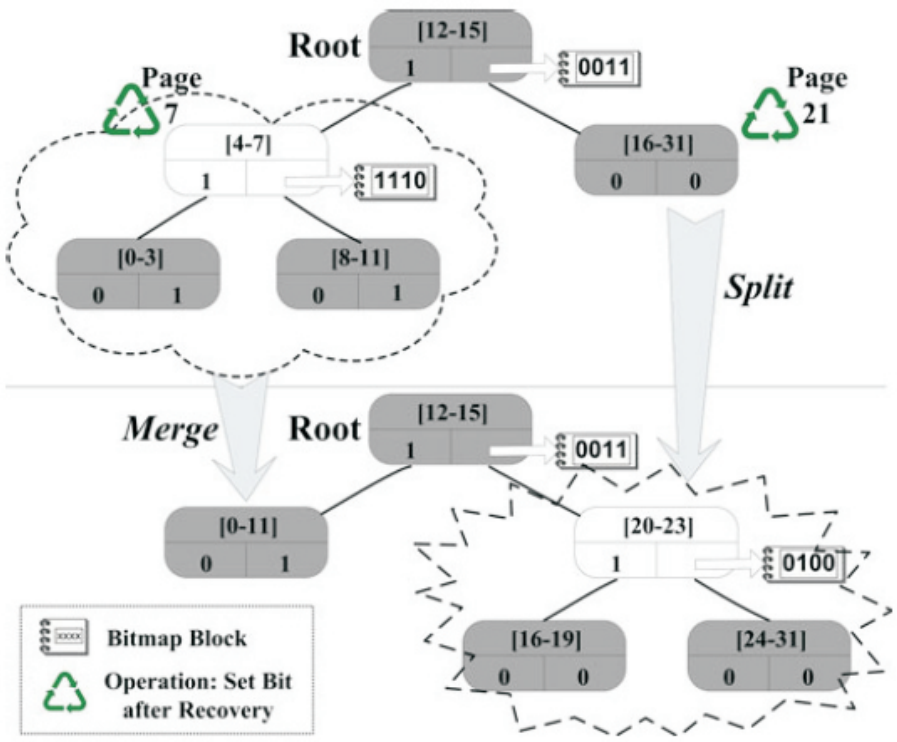


Fig. 4. Red black interval tree based bitmap.

following fields: (a)Range. It is a closed interval represented by its begin and end point. The begin point is the first storage data page index covered by this node, while the end point is the last index. In the tree, all nodes are sorted by the range field. (b) Class. In this bitmap, all nodes are broken down into two classes, distinguished by the lower-left cell of each tree nodes in Fig. 4. One is calledF classand bits in bitmap blocks covered by aF nodemust be all set or all unset simultaneously. In fact, for this kind of node, a flag to store the bit state is enough. The other is called P classand this kind of node are created accompanying with a bitmap block. In BIRDS, aP nodecan only hold one bitmap block which occupies one physical page and can be allocated easily. These two kinds of nodes can convert each other. (c)Value. As shown in the lower-right cell of each tree nodes in Fig. 4, for a F node, value is either '1' or '0' representing the bit state of related bitmap blocks. For aP node, value is a pointer to the bitmap block of this node. (d)Color. A node can be set to either red or black. The principle of color configuration and modification is same as that of standard red black tree.

Two special operations, merge and split, are supported by our red black interval tree based bitmap. For simpleness, we assume each bitmap block contains only 4 bits. As illustrated in Fig. 4, the whole storage space contains 32 pages whose recovery states are recorded in the tree like bitmap. The node whose range is4-7is a P node. From the bitmap block it holds we know that all pages except page 7 are recovered. When page 7 is fetched and the related bit is set, this P nodeconverts to a F nodeand the bitmap block is released. Then, we should try to merge 4-7node with its predecessor and successor, which are its two children in this example. Because these three nodes are allF nodewith bit set now, they can be merged into one node covering all pages in0-11. As shown in the figure's right part, F node with range16-31and bit unset can be split into 3 nodes when page 21 is recovered.P nodewith range20-23is created and inserted into the tree accompanying with the allocating of the related bitmap block, which record the recovered state of page 21.

Determining the Order of Data Retrieval.As described in Section 2.4, the essence of parallel data recovery in BIRDS is fetching data back according to their emergencies. In particular BIRDS uses a predefined pipeline policies: The missing data that blocks the execution is the most urgent data, BIRDS retrieves data immediately upon a data miss in a FIFO fashion. Besides it, it also prefetches data in the background when there is some usable network bandwidth.

The order of data fetch can be defined with the help of two queues: one for emergent data requests, another for prefetching data requests. Data requests in the emergent queue are always processed first before those in the prefetching queue. When data miss happens, the corresponding data recovery request is inserted to the emergent data queue and processed in a FIFO style. When a data request (either from the emergent queue or the prefetching queue) is processed, the requests for its two neighbor data blocks with respect to the physical address are inserted to the prefetching queue too. The rationale behind is that we predict which data will be needed soon based upon sequential locality on the storage. Data in the prefetching queue will not be really processed and recovered until the emergent queue is empty.

Note that we can optimize the data prefetch policy further by using other context-aware knowledge like directory tree localities or heuristic I/O behavior predictions in the future. Such data priority determinate policy is guaranteed bySMAP I/O Analyzerin BIRDS, which is also implemented in the generic block layer. SMAP I/O Analyzerresponses to data miss events generated bySMAP I/O Interceptor, and makes the decision on which data request should be processed next.

Processing Data Requests.TheDispatch Daemondispatches data retrieval requests issued by theSMAP I/O Analyzer. Upon a request, it fetches data from the remote backup data center and writes the received data to local disks. TheDispatch Daemonis implemented in the generic block layer, and mainly uses three data structures: therecovery bitmapmentioned before which records the recovery state of persistent storage data; a retrieval queue containing data retrieval requests; and a dirty queue containing dirty pages that needs to be flushed back to the local disk. It also maintains a page buffer pool in the memory to facilitate the retrieval process.

In BIRDS, all missing data is retrieved in unit of physical page. The actual data retrieval is performed as follows. Upon a request of data retrieval by theSMAP I/O Analyzer, the Dispatch Daemonassigns a free page buffer from the pool to the request. The request is further passed to theData Send & Receive Scheduler. While waiting for the response, the assigned page buffer is pushed into the retrieval queue. When the data is fetched back from the remote site, the recovery bitmapmentioned previous is set to indicate that the corresponding physical page of the persistent storage is now recovered, and the data is written to related page buffer first. Then, the page buffer is removed from the retrieval queue, and pushed into the dirty queue. Finally, the Dispatch Daemoniterates the dirty queue and sends requests to DMA engine to flush the data in these dirty pages into disks. After getting the acknowledgement from the DMA engine, we free the page buffers and the data is recovered successfully.

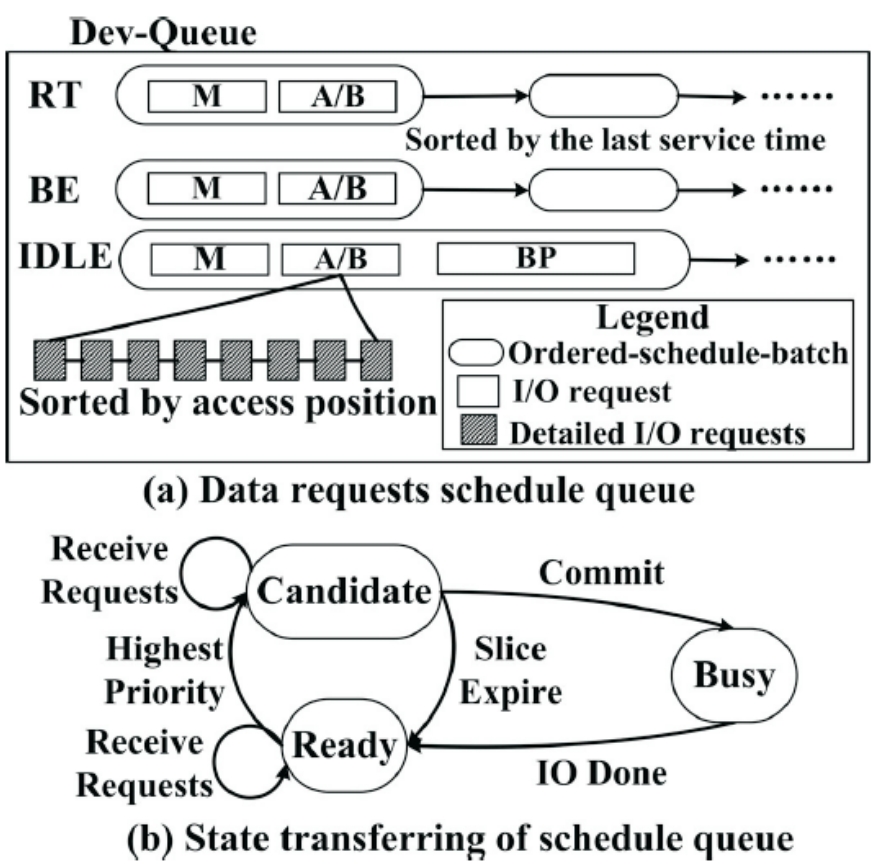


Fig. 5. The scheduler.

The Dispatch Daemon maintains a free page buffer pool instead of using the default I/O page cache management mechanism in OS for two reasons. (1) Dispatch Daemon works in the generic block layer, whereas OS's I/O page caches always exist in a upper layer, such as VFS layer. For the sake of performance, we need a free page buffer pool managed in the generic block layer. (2) The default cache mechanism focuses on reusing cached data pages, which is not the case in data recovery, because same data will never be recovered twice. Hence, our implementation of the page buffer pool does not need to consider such feature and can potentially save some cost. In addition, several features are implemented to lower the cost of page allocation and elease, such as dynamic expansion and contraction of the free page buffer pool, and recycling of page buffers.

Scheduling Data Recovery Process.AData Send & Receive Scheduleris implemented to handle data requests which targets shortening the response time while providing a balance between foreground I/O performance and background data recovery efficiency. We design a data structure that holds variety of I/O requests for the goal. Fig. 5a shows the device queue that is the main buffer chain for a storage device. It is unique for each physical storage device and corresponds to the I/O command queue at the driver layer. All I/O requests are buffered in an order according to their priorities before actually dispatched to the device driver.

To order all I/O requests in the device queue wemainta int hreeseparat epriorityqueuesi nsidetheq ueue to hold I/O requests generated by different types of processesasshowninFig.5a.RT queueholds I/O requests from real time processes such as video applications. IDLE queue holds I/O requests that are issued in an idle context implying that these requests should be served only when no

other requests use the device. All other I/O requests are served with the best effort approach and therefore placed inBE queue.Clearly,we

TABLE 1
EXPERIMENTAL CONFIGURATION PARAMETERS

Experiment Parameters	*Value Range*	*Default*
Network Latency	*10-500ms*	*10ms*
Network Bandwidth	*10-160Mbps*	*80Mbps*
Recovery Block Size	*4-256KB*	*16KB*

determine their priority and place them to corresponding queues according to the importance of I/O requests. We further group I/O requests into ordered-schedule-batches inside the queues, shown as oval boxes in Fig. 5a. Each ordered-schedule-batchcontains I/O requests that share the same characteristics defined by a 2-tuple (I/Oattr,PID) which indicates the same I/O directions and process IDs. These requests will be dispatched or withdrawn together. The ordered-schedule-batches inside each priority queue are resorted based on the two-tuple characteristics and their last service time to avoid starving problem.

I/O requests in each ordered-schedule-batch are then classified into three categories: M-requests, A/B-requests, andBP-requests. M-requestscome from data misses of the running processes which should be given higher priority. We always placeM-requestsat the front of the queue in anordered-schedule-batchas shown in Fig. 5a.A/B requests are generated either by I/O operations from the consistent backup of BIRDS, or by I/Os issued by other application. They are appended next toM-requests. BP-requests are background I/O requests and appear only in IDLE queue after A/B requests. All requests in each orderedschedule-batchare further resorted in a similar way as elevator algorithm used in Linux to minimize disk head movements during I/O operations.

The ordered-schedule-batch can be in one ofthreedifferent statesatruntime: Ready, Candidate,andBusy.The state diagram shown in Fig. 5b illustrates how the state changes overtime. The ordered-schedule-batch is ready as soon as it is added to the priority queues. It becomes a candidatewhen it is selected by the scheduling algorithm as the next one to be dispatched into the device driver. The busy state will be reached if the batch is already dispatched and waiting for I/O responses. The state transitions are easy to follow which accounts for common cases under real world workloads.

4.EVALUATION

We implemented a BIRDS prototype in Linux and preloaded it on a USB drive, which is plugged into a service providing node to make it protected. The protected node is then connected through a Gb Ethernet to a backup server, where different network bandwidths and latencies can be configured through a network emulator for evaluation purpose. We have carried out extensive experiments to test the BIRDS prototype and measure its performance as compared to various baseline methods. Current baremetal DR techniques require full data restore before the server resumes its operations, among which we choose two baseline methods: CDP and a self-implemented traditional DR technique which follows traditional remote disk image replication and cold reboot based recovery mechanism. Table 1 summarizes the experimental configuration

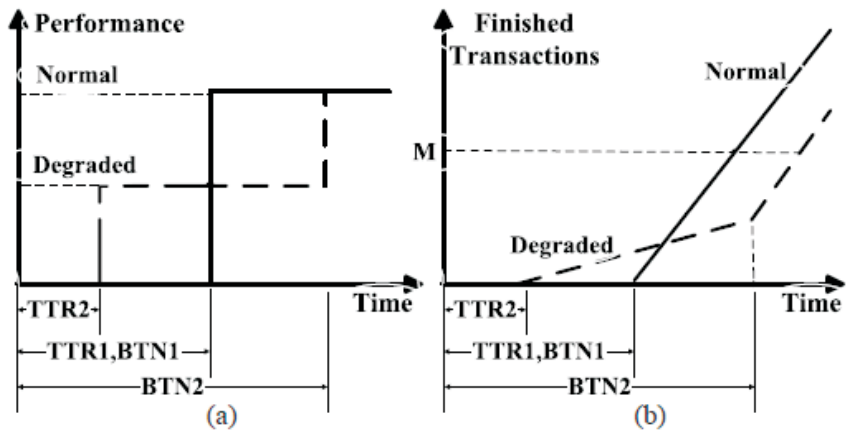


Fig. 6. Example of service degradation. Dashed line indicates a service degradation process. Solid line identifies the normal process.

parameters considered in our experiments. The parameter is set to its default value if notmentioned otherwise.

4.1 EVALUATION METRICS

As a commonly used metric in disaster recovery planning, the recovery time objective[4] specifies the maximum allowable delay until application service is restored after disasters. In another word,RTO indicates the maximum allowable downtime of protected services.time to recovery (TTR) [4] is measured to see whether the predefined RTOhas been achieved after disasters. Compared with it, another frequently used metric is back to normal time (BTN),derivedfrommean time to repair (MTTR)[26]. It means the time required to restore a system to its normal state, i.e., the time needed fortherestorationofnormal service quality, after disasters. For traditional data-oriented DR methods,BTN equalsTTR because the system will be fully restored after all data is recovered.

However,TTR and BTN may lead to contradictory conclusions. Fig. 6a shows a confusing example of the comparison between a traditional recovery process and a recovery process with degradation, in whichBTNfavors the traditional one (i.e., BTN1is smaller than BTN2, whereas TTR suggests the other way around as the recovery with degradation can start serving users earlier. In other words,TTR orBTNalone is not sufficient for evaluating recovery processes with degradation.

Alternatively, we can measure the time to finish a certain amount of service tasks (denoted astime to finish (TTF)) after recovery starts for a more direct comparison. This is analogous to measureTTR with a certain amount of backup data when comparing two traditional data-oriented recovery processes. In another words, when evaluating the performance of a degraded data service process, we could appoint a test set which consists of a fixed number of service tasks. The recovery process that resumes the service to finish the test set with smaller TTFis more efficient. Fig. 6b shows the number of cumulated finished tasks in terms of transactions during the two recovery processes shown in Fig. 6a. The size of the test set is labeled asM. In real DR planning,Mshould be large enough for fair comparison. One possible way is to estimate the total number of user requests in an hour or a day depending on the scale of recovery and setMto match it.

Finally, we propose a novel metric,recovery performability level (RPL), which is specified in terms of average service performability during recovery in proportion to the maximum performability.

Assuming the time duration from the start of recovery to timeTis divided into ntime buckets, and an application dependent performance metricPis measured for each bucket.Pmaxis the maximum measured performance under normal conditions. Then RPLT for the recovery up to timeTcan be calculated as follows:

$$RPL_T = \frac{1}{n}\sum_{t=0}^{n-1}\left(\frac{\mathbf{P(t)}}{\mathbf{P_{max}}}\right)\cdot 100. \qquad (1)$$

Since RPL_{BTN} for traditional DR methods is equal to 0, we useRPLTTF instead for more informative comparisons. Note that sinceTTFis highly related to RPLTTF, we only reportRPLTTFfor evaluation.

In summary, we useTTRtogether withRPLTTF(shorted asRPLbelow) to evaluate the recovery capacity of DR systems fully. We describe the details as follows

- Time To Recovery. The delay until user requests can be fulfilled by the recovered computer system after disasters. We only count data transfer time for traditional DR recovery, eliminate unstable human intervention timeHand service rebuilding timeIduring full recovery process. But for BIRDS, full recovery from bare-metal time are considered.
- Recovery Performability Level. According to Equation (3), the RPLshould be calculated by measuring an application specific performance metric P. For TPC-C test, the metric is defined as the averagetransactions per second (TPS) to match the definition of TPC-C benchmark score. As for SPECWeb test, since original SPECWeb score cannot reflect the instantaneous performanceP(t)for timet, we use the proportion of different service response time range to be the performance metric in SPECWeb test which will be detailed in Section 4.2.2. The performance metric is data transfer rate forVoDapplication, and IO throughput forFTP, DD, GCCandGREPapplications. The time bucket is set to be 5 minutes in our experiments.

4.2 EXPERIMENTAL RESULTS

4.2.1 TPC-C BENCHMARK

In order to run TPC-C benchmark, we installed Oracle 10g DB on a server that runs Ubuntu12.04. The server is configuredwithaCore22.4GhzCPUand2 BRAMandsetup a data warehouse following the TPC-C specifications based on Orabm [27] benchmark. The main functions of TPC-C include inventory control: SERVER_C_SP_Stock_Level, order processing: SERVER_C_SP_Order_Status_ID and SERVER_C_SP_Order_Status_Name. We populate the DB with practical examples to run the main bench, and choose a test set with 10 million transactions for each experiment. BIRDS is plugged into the server and all the transactions and data are backed up every 5 minutes. To test the recoverability of BIRDS, we artificially create a “disaster” which destroyed all 36 GB data in disk and system RAM of the protected server. We then try to carry out bare-metal recovery using two methods: 1) Instance recovery using BIRDS, i.e., as soon as the process checkpoint is available,

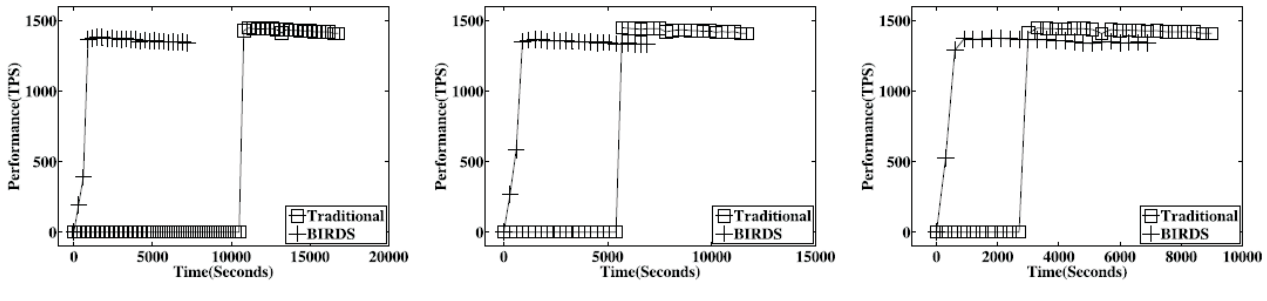


Fig. 7. Sampled process recovering TPC-C benchmark for completing 10 million transactions.

BIRDS restarts TPC-C transactions while data recovery is still in progress; 2) traditional DR recovery method that first recovers both checkpointand persistent data before restarting TPC-C transactions.

Fig. 7 shows the TPC-C performance (in terms of the transactions per second averaged every 15 seconds) when recovered by BIRDS and the traditional DR system with network bandwidths of 40 Mbps, 80 Mbps, and 160 Mbps. Asshowninthisfigure,BIRDScanbringthedataservice back very quickly while the traditional DR technique restarts data service after a long delay for data transfer. TheTTRof BIRDS are 110.31, 87.66, and 76.33 s for network bandwidths of 40 Mbps, 80 Mbps, and 160 Mbps, respectively. The traditional DR technique, on the other hand, cannot restart data service until after 10375.78, 5187.94, and 2595.92 s corresponsively. These service down times may imply significant cost to businesses. It is interesting to note that when the network bandwidth is low, e.g., 40 Mbps, BIRDS finishes all 10 mllion transactions even before the traditional DR system restarts service. This result implies the great potential benefit of BIRDS that exploits data locality property to restart data services using just a small working set of data as opposed to rebuild the entire volume with the traditional DR solutions.

Fig. 8 shows theRPLof BIRDS and the traditional DR system with various network bandwidths. It is clear from this figure that BIRDS provides much higher RPLthan the traditional DR solution. The advantages of BIRDS are greater for lower network bandwidths. For example, when the network bandwidth is about 10 Mbps, theRPLof BIRDS is over five times better than that of the traditional DR

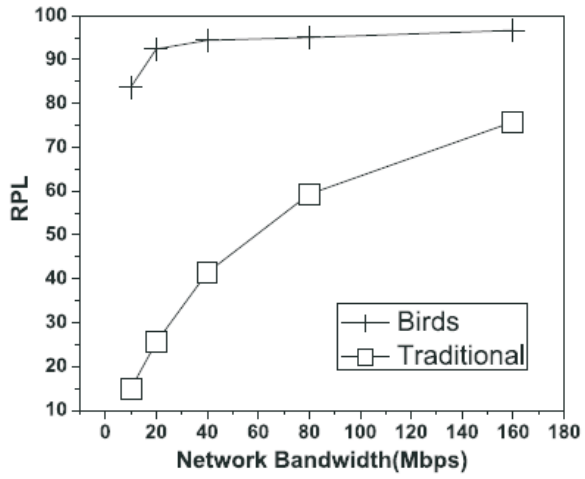


Fig. 8. TPC-C recovery perf. level with varied bandwidths.

solution. This RPL improvement is practically important since 10 Mbps network is equivalent to six to seven T1 lines that are deployed by most small to medium size businesses. In this type of network settings, BIRDS is able to bring back the data service almost instantly with very good service quality during the recovery period. In addition to experiments under different network bandwidths, we carried out additional tests by varying network latencies from 100 ms through 500 ms. Fig. 9 shows the TPC-C performance (in terms of the transactions per second averaged every 15 seconds) when recovered by BIRDS and the traditional DR system with various network latencies. As shown, BIRDS restarts data service much earlier than the traditional DR solution for all latencies considered although BIRDS' performance changes with respect to different network delays. The RPL is shown in Fig. 10 as a function of latency for both BIRDS and the traditional DR solution. It should be noted that the performance of the traditional DR solution stays almost the same for all network latencies considered. This is because our experiments assume that the traditional DR solution continuously transfer data from the backup site to the production site during data recovery period.

The network latency only affects the first data packet and all the following data packets are transferred continuously in a pipelined fashion. As a result, a few hundreds of milliseconds delay in the beginning of data recovery does not show any significance in the total data recovery time. In contrary, BIRDS retrieves data with some randomness which results in different RPL values for different latencies as shown in Fig. 10. Nevertheless, BIRDS shows much better RPL than the traditional DR solution across all latencies tested.

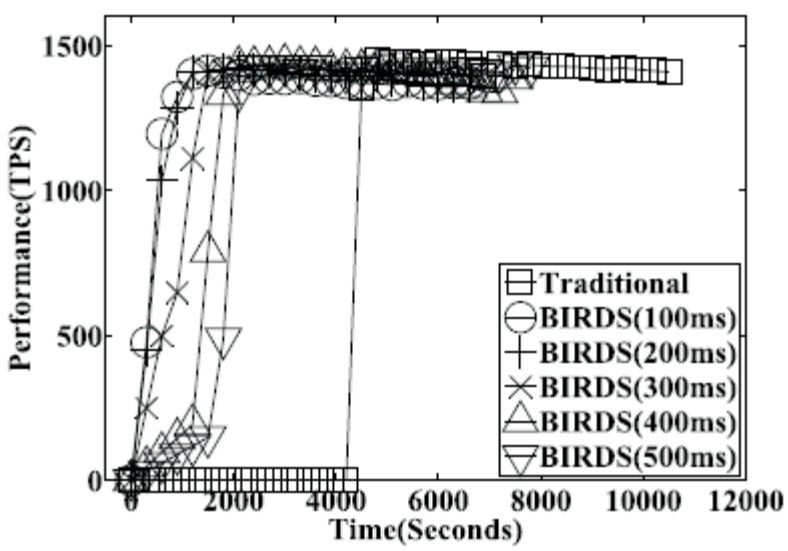


Fig. 9. Sampled recovery process for TPC-C with varied latencies.

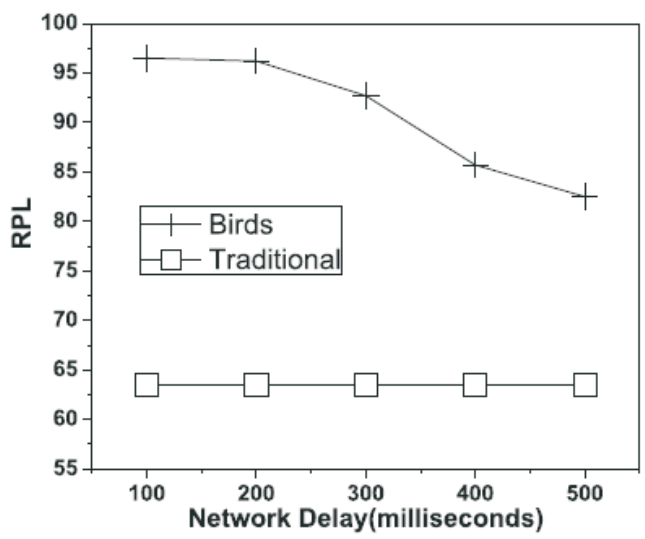


Fig. 10. TPC-C recovery perf. level with varied latencies.

4.2.2 SPECWEB BENCHMARK

We use SPECWeb 2005 for the test. Due to page limit, we only show "Banking" workload in this section. Results for "Ecommerce" and "Support" are similar. Four PC servers interconnected using a Gb Ethernet are used in this experiment. One PC is used to generate web requests on behalf of four clients and the other three PCs act as web server, BIRDS backup server, and BESim simulator, respectively. The web server here is the BIRDS protected server, it is configured with a Xeon 2.33 Ghz CPU, 16 GB RAM, installing a Linux Ubuntu12.04 and php-fpm/nginx applications. The running states and data of the web server are backed up every 5 minutes. The BESim simulates the web service backend and is in charge of database operations.

The performance score of SPECWeb cannot be used directly to measure the performance level of the resumed service in our experiments during recovery. Actually, the performance score of SPECWeb is obtained by varying the workload of web requests, and set to be the maximum number of "simultaneous sessions" under which the web server can still response within reasonable time. In particularly, if a web page is correctly returned within 2 seconds after a request, it is considered as GOOD, 4 seconds as TOLERABLE, and not in time as ERROR. The final reported score is determined by the maximum total number of concurrent web connections requested by clients, among which at least 95 percent are GOOD, 99 percent are TOLERABLE, and less than 1 percent are ERROR.

In order to measure how the performance level of the resumed service increases after recovered by BIRDS, we need a performance metric that can be measured at each time point. Apparently, the performance score of SPECWeb described above is hard to fulfil such requirement. Alternatively, we measure the respective percentages of GOOD, TOLERABLE, and PASS (¼100%-ERROR) requests as performance metrics at each time point during the recovery period using a specified workload. Repetitive tests show that our web server's SPECWeb score is 717. In other words, it can handle maximally 717 concurrent sessions normally and get a GOOD percent of nearly 100 percent. In order to illustrate the effects of varying bandwidth and latency more clearly in our experiments, we choose to use a heavier workload than 717 concurrent sessions to measure the performance metrics during RECOVERY. That is

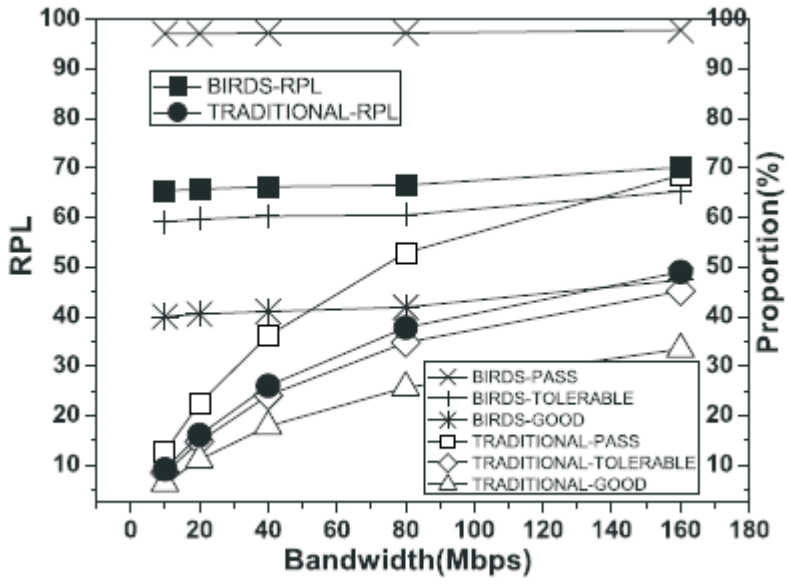


Fig. 11. SPECWeb recovery perf. level with varied bandwidths.

a workload of 1077 concurrent sessions (i.e., 150 percent of 717 concurrent sessions) was used for our experiments. We measure the respective percentages of GOOD, TOLERABLE, and PASS for both the normal system and resumed system, and calculateRPLs accordingly. Furthermore, we also average GOOD, TOLERABLE, and PASSRPLstoget an overallRPLfor our tests.

With such method, we measure theRPLof SPECWeb during the recovery period and the results are shown in Fig. 11 as a function of network bandwidth. It can be seen from this figure that BIRDS provides much better RPL than the traditional DR solution. During the recovery period, BIRDS restarts data service quickly and provides over 95 percent of PASS requests. About 40 to 50 percent of requests are GOOD and 60 to 70 percent of requests are TOLERABLE. The traditional DR solution, on the other hand, performs poorly giving the percentage of PASS from less than 10 to about 70 percent depending on the network bandwidth. It is interesting to note that BIRDS's RPLdoes not change greatly with different network bandwidths implying the effectivness of the network volume during recovery period. Similar to TPC-C experiments, we have also tested SPECWeb under different network latencies as shown in Fig. 12. We can see that BIRDS's RPLgradually reduces as network latency increases. We observed again that theRPLof the traditional DR solution does not change with network latency for the same reason explained previously. Nevertheless, the percentages of GOOD, TOLERABLE, and PASS of BIRDS are all much higher than those of the traditional DR solution.

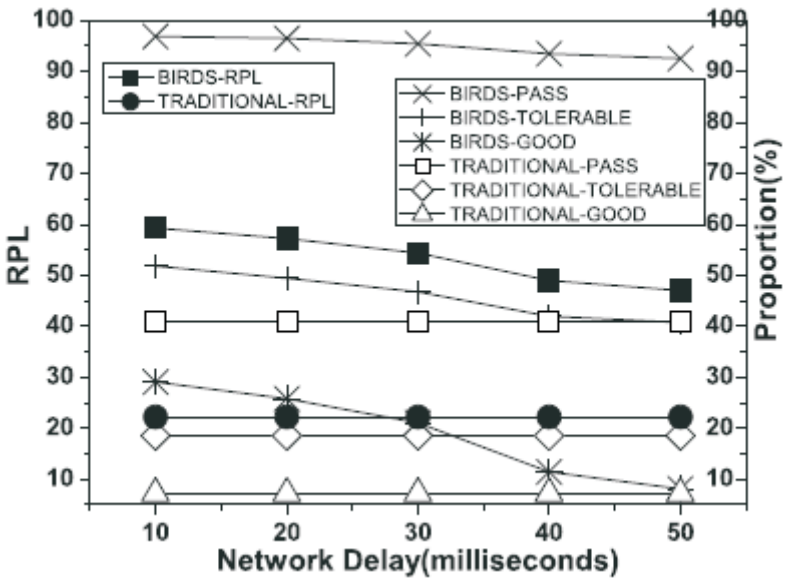


Fig. 12. SPECWeb recovery perf. level with varied latencies.

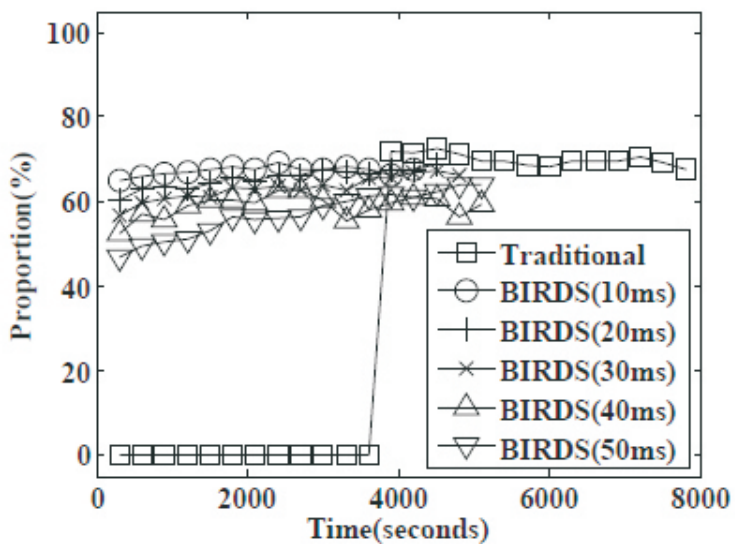


Fig. 13. Recovery process of SPECWeb with varied latencies.

For the page limit, we cannot show full recovery processes in different network conditions. As an example, the process when recovering SPECWeb with a 80 Mbps network with varied latencies is shown in Fig. 13. TheTTRof BIRDSisabout30secondswhichmeansdataserviceswill be instantly usable after BIRDS is rebooted, while it will be about 4,000 seconds using traditional DR method. We only test with latencies from 10 to 50 ms, because the metrics measured in SPECWeb such as GOOD, PASS and TOLERABLE is latency sensitive. Besides the smallTTR measured with BIRDS, we can see the performance is increased slowly as the recovery process advances, reflecting less page fault when more data have been restored while the recovery proceeds.

4.2.3 BIRDS VERSUS CDP

We use a FTP service to show the difference between Birds and CDP. Many systems only use CDP locally for data rollback. But for DR support, geographically separated remote CDP site must be deployed. As a typical example, FalconStor continuous data protector implements CDP combined with remote data replication to do this. For the lack of commercial CDP software, we implement a CDP tool here. The CDP tool records the data updates and transfers them toaremotebackupsitecontinuously. We use both Birds and the CDP tool to protect a FTP server respectively and replay a FTP log in the protected FTP server. The log is collected from a heavy-loaded FTP server on our campus, including 57,366 sessions in 2.5 hours and 80 GB data involved. To simulate lots of data updates which need to be protected by CDP or Birds, we reverse the data flow

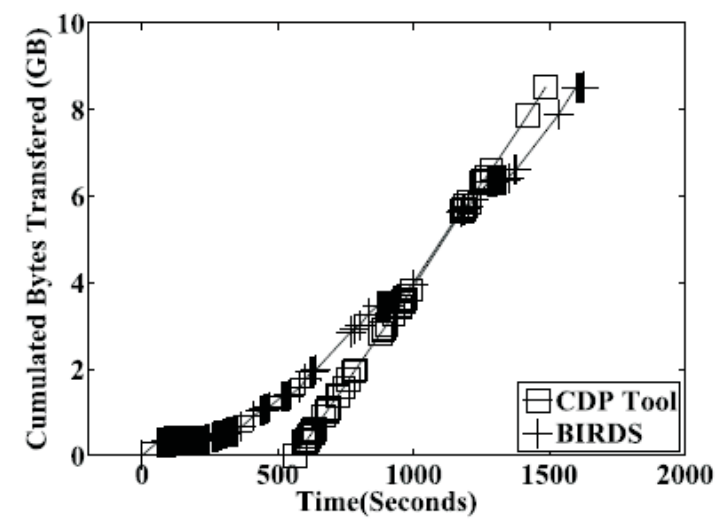


Fig. 14. Performance for recovering to checkpoint 7200s.

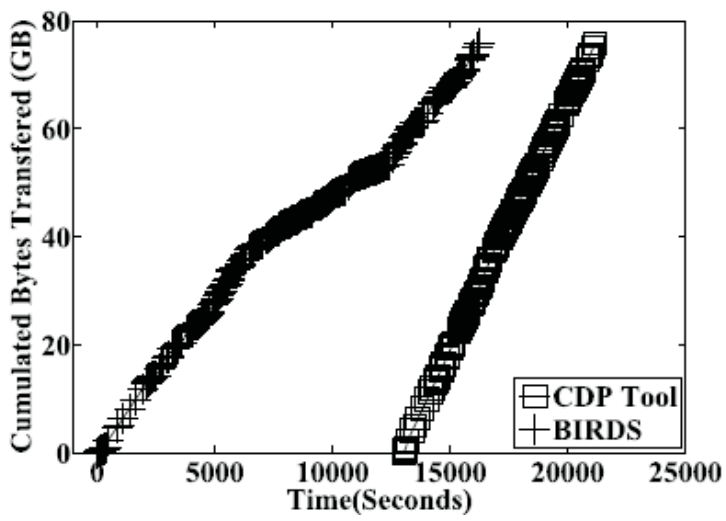


Fig. 15. Performance for recovering to checkpoint 0s.

and set all sessions to upload sessions because original download data flow do not change any data in

the server. The bandwidth of the interconnect network is 100 Mbps and the latency is set to 100 ms. We artificially create five clean checkpoints across the 8,132 second long replay duration. The checkpoints are set at 0, 1,800, 3,600, 5,400, 7,200 s respectively. A man-made disaster is generated to crash the service after 7,500 seconds. After the disaster, we tried to recover the FTP service to the five checkpoints by Birds and CDP separately, and continue to finish the replay of the log. As an example, the recovery processes back to two different time points of replay are demonstrated in Figs. 14 and 15.

We can see that Birds can restore the service almost at once (about 0.23 second after the start of recovery). For the last checkpoint (7,200 s point), the CDP tool resumes the service 563.16 seconds later than Birds, but finishes the job 141.34 seconds ahead of Birds. For earlier checkpoint, Birds outperforms the CDP tool completely. When we try to recover to the first checkpoint (0 s point), Birds resumes the service 13027.43 s ahead of the CDP tool, and completes the whole task 4886.72 s earlier. The detailed performance differences between Birds and CDP can be clearly seen in Fig. 16 in terms of RPL. We can conclude that Birds outperforms CDP as the gap between failure time point and recovery time point increases.

4.3 Runtime Overhead of BIRDS

Backup Overhead.BIRDS needs to backup the persistent storage of protected computer systems to a geographically remote site periodically. In the protected production site,

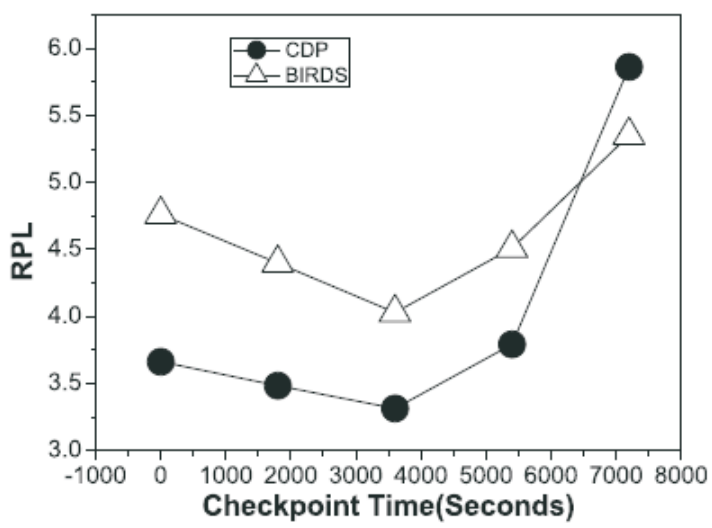


Fig. 16. The average throughout ratio of Birds over CDP.

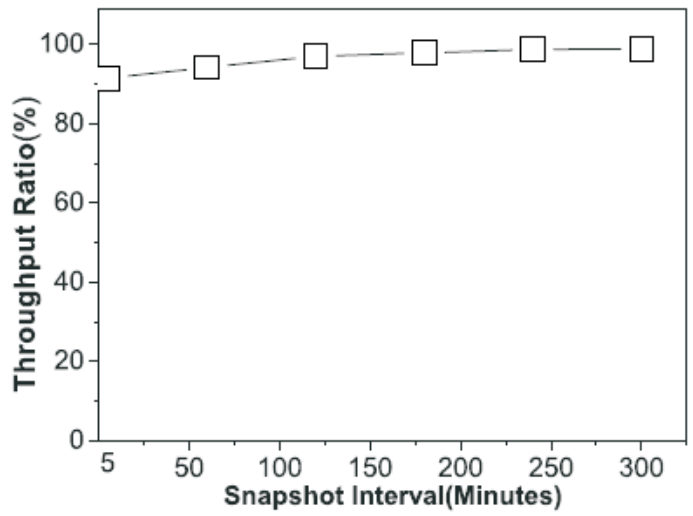


Fig. 17. Snapshot Overhead Measurement.

backup overhead is introduced by snapshot mechanism which traces and records data changes. In order to estimate the overheads caused by BIRDS, we run the TPC-C benchmark for 1 million transactions and set a backup interval from 5 minutes to 300 minutes. The performances (TPS) were measured on the system without BIRDS and the system with BIRDS installed. We then compare the performances of the two systems as shown in Fig. 17, the overall performance difference ranges from 91.5 to 98.86 percent, which indicates acceptable snapshot overheads from 1.14 to 8.5 percent. The overhead increases with smaller snapshot interval due to full replication of the running states, which can be further improved.

In the TPC-C test, the protected machine is configured with 2 GB RAM. During each backup, on average 53 and 1.23 MB data is transferred for saving memory states and incremental persistent storage data, respectively, when the backup interval is set to be 5 minutes. The backup process continues for about 2 seconds.

OS virtualization Overhead.As illustrated in Fig. 1, an OS virtualization layer is introduced between services and the low level disk drivers, which brings additional overhead. We evaluate the IO throughput overhead of OS virtualization using IOMeter benchmark. We use one IOMeter worker running to generate random and sequential workloads with the request size of 4 KB. The result is shown in Fig. 18. The overhead brought by OS virtualization only ranges from 0.1 percent to 0.2 percent with the increase of random access proportions.

The overheads introduced by BIRDS are also measured by running TPC-C and SPECWeb with BIRDS installed

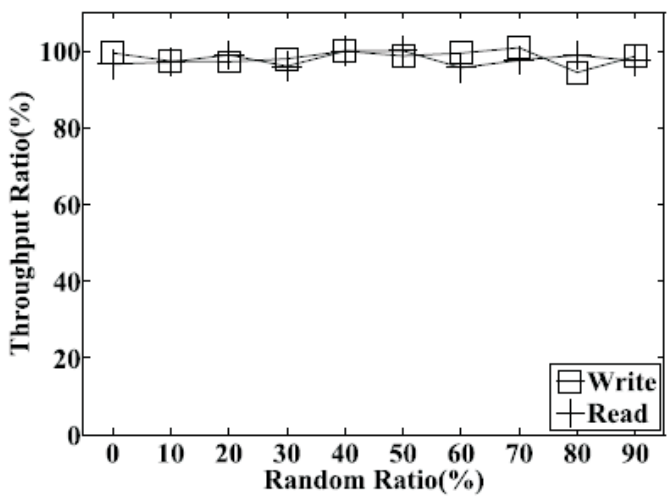


Fig. 18. Overhead Introduced by OS Virtualization.

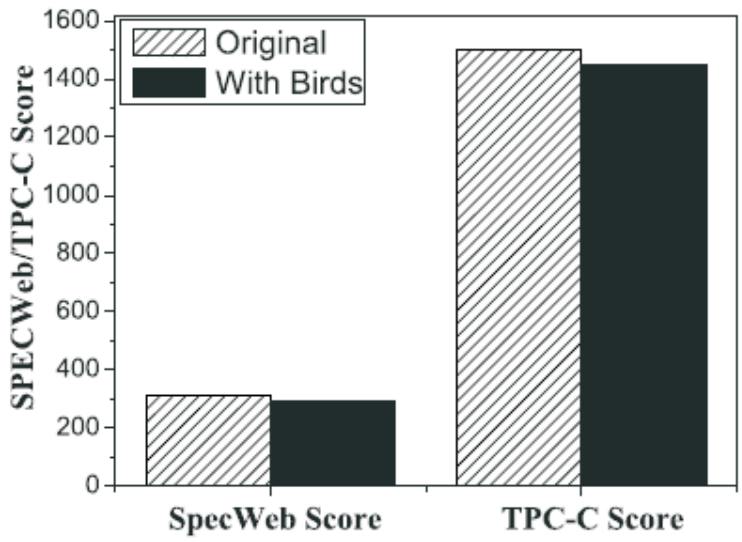


Fig. 19. Total Overhead Shown in TPC-C SPECWed Test.

compared to a system without BIRDS under normal working conditions. Fig. 19 plots the performance comparisons of the two systems for the two benchmarks. As can be seen from the figure, the overhead is quite small, about 3.3 percent in TPC-C and 4.5 percent in SPECWeb.

4.4 OTHER PERFORMANCE FACTORS

In this section, we will exploit the impact factor influencing BIRDS performance. To begin with, we consider datafetching policy and the recovery block size that means the basic data unit for BIRDS to fetch data from the remote backup data center. In our implementation, the recovery block size must be multiples of the physical page size, i.e., 4 KB. We use BIRDS to recover a FTP service which is dealing with thousands of file requests according to a collected log. We use two data fetching policies here. One is the pipeline policy used by BIRDS, the other is on-demand policy which eliminates the background prefetch of pipeline policy.

By varying the recovery block size and data fetching policy, we get the result in Fig. 20. It is clear that no matter what recovery block size is, the pipelined data fetching policy always outperforms the one using on-demand fetching due to the benefit brought by the background prefetch. Most importantly, the recovery performance is not increasing steadily with larger recovery block size. The maximum performance appears when the recovery block size is set to 128 KB in this case. The main cause is that more interrupt and freeze will happen when the block size is too small, while too large blocks will result in the lost of the potential parallel characteristics of recovery process either.

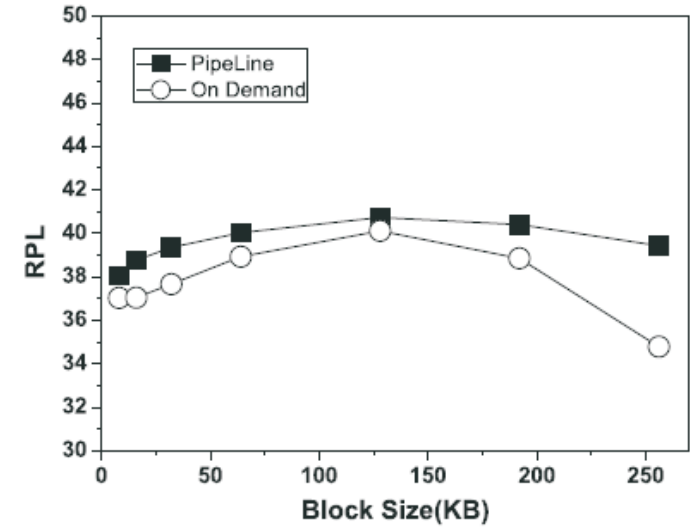


Fig. 20. Performance change with varied recovery block size.

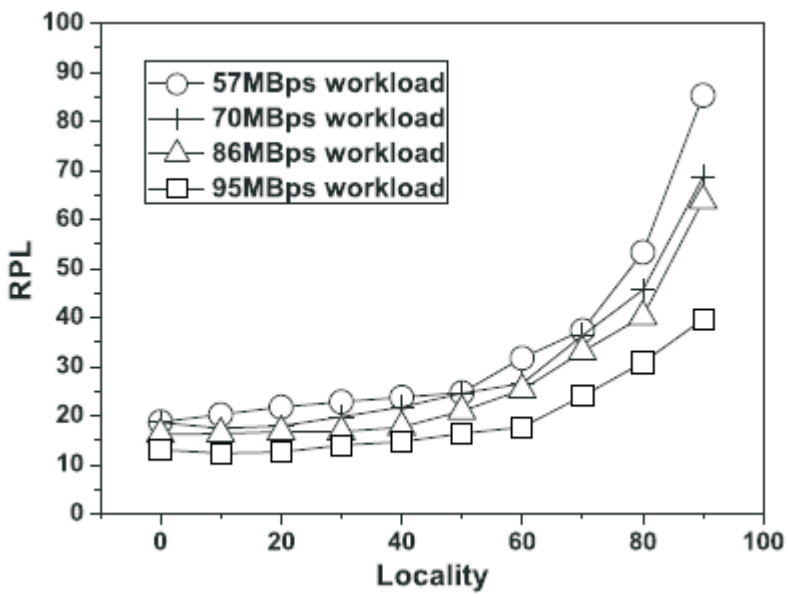


Fig. 21. Recovery performance for data replication.

Second, we design and implement a toolkit based ondd (a common Unix program whose primary

purpose is the low-level copying and conversion of raw data). It is used to simulate applications with different IO characteristics in terms of data locality and IO tension level. The locality here is defined by the ratio between repeatedly accessed data amount and total accessed data amount. Workloads with varied IO tension levels are got by manipulating the replicated data block size used bydd. Each test uses a workload consists of mixed read-write operations on totally 64 GB data with configurable IO tension level and data locality. The ratio between read and write operations is set to 1 : 4 to reflect typical file system behaviors. As shown in Fig. 21, the recovery performance increases with increased data localities and decreased IO tension level.

It is easy to understand that data localities benefit the parallel recovery process of BIRDS because fewer remote data fetchings will be needed with more repeatedly accessed data restored before. Moreover, it is also reasonable to guess that lighter IO pressure benefits the parallel recovery process too. To verify the conclusion, we repeat the test using BIRDS to restore systems which is running different applications with different IO tensions. Three tasks are tested: (1) Compiling a Linux-2.6 kernel byGCC. (b) Downloading nearly 16 GB data by SFTPclient. © UsingGREPto find string patterns throughout a file system which contains approximately 40 GB data. We modify the SMAP interceptor to trace resulted IO behaviors andestimateresultedIOtensions(inthenameofIO throughput) and localities by static analysis on gathered logs. The recovery performance are plotted in Fig. 22. We can see that with lighter IO tension and better data locality, better recovery performance can be expected.

5.RELATED WORK

Having realized the critical importance of disaster recovery to business continuity, many DR solutions have emerged recently both in terms of storage products and research reports. To deal with the increasing complexity of DR designs, an automatic tool [4] was proposed for storage administrators to select appropriate designs to meet their performance, cost, and reliability objectives.

Traditional Data-oriented Methods.Traditional DR solutions focused on backup and recover persistent storage data for disasters. During each backup-and-recovery

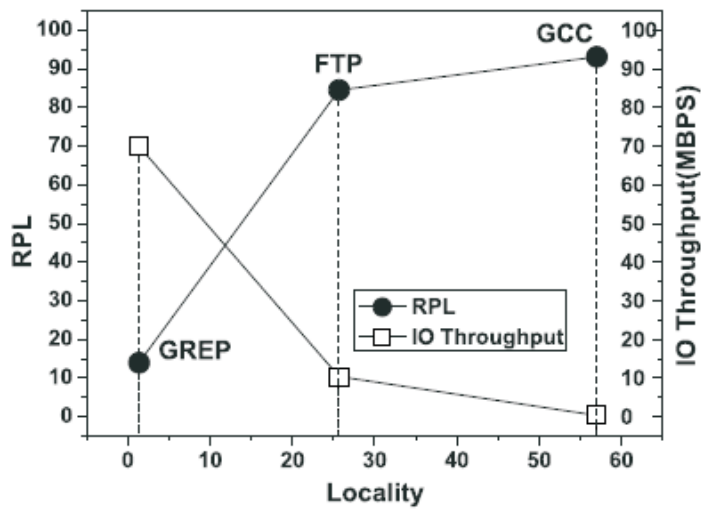


Fig. 22. Performability comparison for varied applications.

cycle, an application consistent recovery point must be saved, found and resumed to ensure the correctness of recovered data services [12].If the backup operations are triggered periodically, the

application consistency is ensured in each backup operations by using applicationdependent methods.

On the other hand, if the backup operations are triggered by data write events or continuously which means data replications will be much more frequent than periodicalbackup, the consistency point cannot be ensured in the backup stage for the avoidance of significant overheads. That is what is done by continuous data protection [13], [14], [15], [16], [17] techniques. A tough task for CDP is to find an application consistent recovery point out of nume rousrecoverypo ints[16].Birdsusesanappli cationindependent way to solve the problem by replicating both the persistent data and the running states simultaneously. Besides, Birds can finish recover a computer system from the scratch without any human interventions, while traditional DR methods suffered from unavoidable manual operations including OS rebooting, processes restarting, and reconfiguring.

Failover Methods. Failover methods use the remote backup site as a hot standby and synchronize the states changes between production site and remote backup site. Again, some methods choose to synchronize only the persistent storage states. Such synchronization systems are commonly classified as data mirror. For example, Snapmirror [8], Seneca [9] and SMFS [10] are remote-mirroring systems for disaster recovery providing good performance and predictable behavior. Existing remote-mirroring DR solutions focus on guaranteeing correct ordering of mirrored data.

Other failover approaches synchronize not only the persistent storage states but also the running states to the remote backup site. Traditionally, the remote backup site need to be constructed with the same infrastructure and hardware with the production site. But recent studies explore virtualization based whole system replication techniques like Remus [29], [30] which used Xen migration to provide high availability by implementing fine-grained remote checkpoints. Dejaview [31] is also essentially the kind of system though designed for recording and replaying the historical status of a computer system. Other related techniques include emerged virtualization based migration techniques [11], [18], [20], [24], [25], [29], [30], [32], [33], [34], [36], [37] that were initially designed for easy management, load balancing, and high availability. Such techniques may also be used to minimize service downtime by failover to the secondary site after system failures.

Whole system failover method is quite different from the whole system failback approach presented in this paper, where running states are suspended, copied, and then resumed instead of switching to another place when encountered with failures.

VM based Backup-and-recovery Methods.VMware consolidated backup (VCB) [19] enables automatic recovery as Birds. But it cannot accomplish the goal of fast recovery after corruptions. The novelty of Birds lies in separating and parallelizing the execution of data retrieval and service rebuilding procedure by using container-based OS virtualization method in a backup-and-recovery system for instant restoration of data services after disasters.

Our system is similar to Collective system [18] and Veeam backup & replication to some extent. Both systems used an on-demand data fetching strategy similar to Birds, but no data prefetching was employed. Moreover, Birds is designed to restore any generic computer system from baremetal, while those systems can only run in a VM supported environment. Container-based virtualization used in Birds has much less overhead than those systems.

Other related systems include virtual machine based time travel system [17], [35] which periodically saving both VM memory states and storage states to disks. The major difference of Birds from them is the emphasis on parallel recovery, and Birds's capability of automatical recovery from bare-metal.

As a whole, to the best of our knowledge, none of the related work mentioned above resolves this issue of instant restart of data services from bare-metal.

6.CONCLUSIONS

We present a new data protection and recovery technique called Birds emphasizing on instant restoration of data services after a failure event from “bare-metal”. Birds backups and replicates not only the persistent storage data but also the point-in-time process checkpoints to ensure application consistency of backup data. After a failure event, Birds can bring up data service back very quickly while data recovery is still in progress. A Birds prototype has been built in Linux with a container based virtualization method. The prototype can be easily plugged into any server to be protected without requiring reinstallation of the server OS. Using the Birds prototype, we have carried out extensive experimental test to measure the performability, recovery time, and run time overheads of Birds. Our experimental results show that Birds can restart application services several orders of magnitude faster than traditional DR techniques after a disaster event. The average performability of Birds during recovery period is as high as 6 times more than traditional DR solutions. Furthermore, the runtime overhead introduced by Birds during normal operations is negligible.

ACKNOWLEDGMENTS

This paper was supported by the the National High Technology Research and Development Program of China(2012AA012600) and the National Natural Science Foundation of China (60973143)

REFERENCES

[1] B.J. Landry and M.S. Koger, “Dispelling 10 Common Disaster Recovery Myths: Lessons Learned from Hurricane Katrina and Other Disasters,”ACM J. Educational Resources in Computing, vol. 6, no. 4, p. 6, 2006.

[2] L.N. Bairavasundaram, G.R. Goodson, S. Pasupathy, and J. Schindler, “An Analysis of Latent Sector Errors in Disk Drives,” Proc. 2007 ACM SIGMETRICS Int’l Conf. Measurement and Modeling of Computer Systems (SIGMETRICS ’07),2007.

[3] B. Schroeder, S. Damouras, and P. Gill, “Understanding Latent Sector Errors and How to Protect Against Them,”Proc. Eighth USENIX Conf. File and Storage Technologies (FAST ’10), 2010.

[4] K. Keeton, C. Santos, D. Beyer, J. Chase, and J. Wilkes, “Designing for Disasters,”Proc. Eighth USENIX Conf. File and Storage Technologies (FAST ’04), 2004.

[5] S. Nath, H. Yu, P.B. Gibbons, and S. Seshan, “Subtleties in Tolerating Correlated Failures in Wide-area Storage Systems,”Proc. Third Conf. Networked Systems Design & Implementation—Volume 3 (NSDI ’06), 2006.

[6] D.M. Smith, “The Cost of Lost Data,” J. Contemporary Business Practice, vol. 6, no. 3, 2003.

[7] K. Keeton, D. Beyer, E. Brau, A. Merchant, C. Santos, and A. Zhang, “On the Road to Recovery: Restoring Data after Disasters,” Proc. First ACM SIGOPS/EuroSys European Conf. Computer Systems (Eurosys ’06), 2006.

[8] H. Patterson, S. Manley, M. Federwisch, D. Hitz, S. Kleiman, and S. Owara, “SnapMirror: File System Based Asynchronous Mirroring for Disaster Recovery,”Proc. Eighth Conf. File and Storage Technologies (FAST ’02), 2002.

[9] M. Ji, A. Veitch, and J. Wilkes, “Seneca: Remote Mirroring Done Write,”Proc. USENIX Ann. Technical Conf., 2003.

[10] H. Weatherspoon, L. Ganesh, T. Marian, M. Balakrishnan, and K. Birman, “Smoke and Mirrors: Reflecting Files at a Geographically Remote Location without Loss of Performance,”Proc. Seventh Conf. File and Storage Technologies (FAST ’09), 2009.

[11] A. Mashtizadeh, E. Celebi, T. Garfinkel, and M. Cai, “The Design and Evolution of Live Storage

Migration in VMware ESX,"Proc.2011 USENIX Conf. Ann. Technical Conf., 2011.
[12] M. Vrable, S. Savage, and G.M. Voelker, "Cumulus: Filesystem Backup to the Cloud,"ACM Trans. on Storage (TOS), vol. 5, no. 4, p. 14, 2009.
[13] G. Laden, P. Ta-Shma, E. Yaffe, M. Factor, and S. Fienblit, "Architectures for Controller Based CDP," Proc. Fifth USENIX Conf. File and Storage Technologies (FAST'07), 2007.
[14] N. Zhu and T. Chiueh, "Portable and Efficient Continuous Data Protection for Network File Servers,"Proc. 37th Ann.IEEE/IFIP Int'l Conf. Dependable Systems and Networks (DSN'07), 2007.
[15] M.K. Aguilera, K. Keeton, A. Merchant, K. Muniswamy-Reddy, and M. Uysal, "Improving Recoverability in Multi-tier Storage Systems,"Proc. 37th Ann. IEEE/IFIP Int'l Conf. Dependable Systems and Networks (DSN'07), 2007.
[16] A. Verma et al., "SWEEPER: An Efficient Disaster Recovery Point Identification Mechanism,"Proc. Sixth USENIX Conf. File and Storage Technologies (FAST'08), 2008.
[17] P. Ta-Shma, G. Laden, M. Ben-Yehuda, and M. Factor, "Virtual Machine Time Travel Using Continuous Data Protection and Checkpointing,"Proc. Haifa Systems and Storage Conf., 2007.
[18] C.P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M.S. Lam, and M. Rosenblum, "Optimizing the Migration of Virtual Computers," Prof. Fifth Symp. Operating Systems Design and Implementation (OSDI'02), 2002.
[19] VMware Consolidated Backup, Enable Non-Disruptive Backup for Virtual Machines, http://www.vmware.com/files/ pdf/consolidated_backup_datasheet.pdf, 2009..
[20] M.R. Hines and K. Gopalan, "Post-Copy Based Live Virtual Machine Migration Using Adaptive Pre-paging and Dynamic Self-Ballooning," Proc. 2009 ACM SIGPLAN/SIGOPS Int'l Conf. Virtual Execution Environments (VEE'09), 2009.
[21] S. Osman, D. Subhraveti, G. Su, and J. Nieh, "The Design and Implementation of Zap: A System for Migrating Computing Environments," Prof. Fifth Symp. Operating Systems Design and Implementation (OSDI'02), 2002.
[22] O. Laaden and J. Nieh, "Transparent Checkpoint-Restart of Multiple Processes on Commodity Operating Systems,"Proc. USENIX Ann. Technical Conf., 2007.
[23] S. Soltesz, H. Potzl, M.E. Fiuczynski, A. Bavier, and L. Peterson, "Container-Based Operating System Virtualization: A Scalable, High-Performance Alternative to Hypervisors,"Proc. Second ACM SIGOPS/EuroSys European Conf. Computer Systems (Eurosys'07), 2007.
[24] M. Nelson, B. Lim, and G. Hutchins, "Fast Transparent Migration for Virtual Machines," Proc. Ann. Conf. USENIX Ann. Technical Conf. (ATEC'05), 2005.
[25] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live Migration of Virtual Machines,"Proc. Second Conf. Symp. Networked Systems Design & Implementation—Volume 2 (NSDI'05), 2005.
[26] J.F. Meyer, "On Evaluating the Performability of Degradable Computing Systems,"IEEE Trans. Computers, vol. 29, no. 8,pp. 720-731, Aug. 1980.
[27] Orabm, "Oracle OLTP benchmarking suite, http://www.linxcel. co.uk/software_orastress.html, 2014.
[28] H. Yu, D. Zheng, B.Y. Zhao, and W. Zheng, "Understanding User Behavior in Large-Scale Video-On-Demand Systems,"Proc. ACM SIGOPS/EuroSys European Conf. Computer Systems (Eurosys'06),2006.
[29] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield, "Remus: High Availability via Asynchronous Virtual Machine Replication,"Proc. Fifth UNISEX Symp. Networked Systems Design & Implementation (NSDI'08), 2008.
[30] U.F. Minhas, S. Rajagopalan, B. Cully, A. Aboulnaga, K. Salem, and A. Warfield, "RemusDB:

Transparent High Availability for Database Systems,"The VLDB Journal, vol. 22, no. 1, pp. 29-45, 2013.
[31] O. Laadan, A. Baratto, D.B. Phung, S. Potter, and J. Nieh, "DejaView: A Personal Virtual Computer Recorder,"Proc. 21st ACM SIGOPS Symp. Operating Systems Principles (SOSP '07), 2007.
[32] R. Bradford et al., "Live Wide-Area Migration of Virtual Machines Including Local Persistent State,"Proc. Third Int'l Conf. Virtual Execution Environments (VEE '07), 2007.
[33] W. Huang, J. Liu, M. Koop, B. Abali, and D. Panda, "Nomad: Migrating OS-Bypass Networks in Virtual Machines,"Proc. Third Int'l Conf. Virtual Execution Environments (VEE '07), 2007.
[34] S. Kumar and K. Schwan, "Netchannel: A VMM-Level Mechanism for Continuous, Transparent Device Access during VM Migration," Proc. Third Int'l Conf. Virtual Execution Environments (VEE),2008.
[35] A. Warfield, R. Ross, K. Fraser, C. Limpach, and S. Hand, "Parallax: Managing Storage for a Million Machines,"Proc. 10th Conf. Hot Topics in Operating Systems—Volume 10 (HOTOS '05),2005.
[36] H.A. Lagar-Cavilla, J.A. Whitney, A. Scannell, P. Patchin, S.M. Rumble, E. de Lara, M. Brudno, and M. Satyanarayanan, "SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing,"Proc. Fourth ACM European Conf. Computer Systems (Eurosys '09), 2009.
[37] S. Al-Kiswany, D. Subhraveti, P. Sarkar, and M. Ripeanu,"VMFlock: Virtual Machine Co-Migration for the Cloud,"Proc. 20th Int'l Symp. High Performance Distributed Computing (HPDC '11), 2011.
[38] Virtualizing Performance-Critical Database Applications in VMware vSphere, http://www.vmware.com/pdf/Perf_ESX40_Oracle-eval.pdf, 2012.