CloudNet: Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines

Timothy Wood, K. K. Ramakrishnan, *Fellow, IEEE*, Prashant Shenoy, *Fellow, IEEE, Senior Member, ACM*, Jacobus Van der Merwe, Jinho Hwang, Guyue Liu, and Lucas Chaufournier

***Abstract*—Virtualization technology and the ease with which virtual machines (VMs) can be migrated within the LAN have changed the scope of resource management from allocating resources on a single server to manipulating pools of resources within a data center. We expect WAN migration of virtual machines to likewise transform the scope of provisioning resources from a single data center to multiple data centers spread across the country or around the world. In this paper, we present the CloudNet architecture consisting of cloud computing platforms linked with a virtual private network (VPN)-based network infrastructure to provide seamless and secure connectivity between enterprise and cloud data center sites. To realize our vision of efficiently pooling geographically distributed data center resources, CloudNet provides optimized support for live WAN migration of virtual machines. Specifically, we present a set of optimizations that minimize the cost of transferring storage and virtual machine memory during migrations over low bandwidth and high-latency Internet links. We evaluate our system on an operational cloud platform distributed across the continental US. During simultaneous migrations of four VMs between data centers in Texas and Illinois, CloudNet's optimizations reduce memory migration time by 65% and lower bandwidth consumption for the storage and memory transfer by 19 GB, a 50% reduction.**

***Index Terms*—Cloud computing, virtualization, wide area network (WAN) migration.**

I. Introduction

TODAY'S enterprises run their server applications in data centers, which provide them with computational and storage resources. Cloud computing platforms, both public and private, provide a new avenue for both small and large enterprises to host their applications by renting resources on demand and paying based on actual usage. Thus, a typical enterprise's IT services will be spread across the corporation's data centers as well as dynamically allocated resources in cloud data centers.

From an IT perspective, it would be ideal if both in-house data centers and private and public clouds could be considered as a flexible pool of computing and storage resources that are seamlessly connected to overcome their geographical separation. The management of such a pool of resources requires the ability to flexibly map applications to different sites as well as the ability to move applications and their data across and within pools. The agility with which such decisions can be made and implemented determines the responsiveness with which enterprise IT can meet changing business needs.

Virtualization is a key technology that has enabled such agility *within* a data center. Hardware virtualization provides a logical separation between applications and the underlying physical server resources, thereby enabling a flexible mapping of virtual machines (VMs) to servers in a data center. Furthermore, virtual machine platforms support resizing of VM containers to accommodate changing workloads as well as the ability to live-migrate virtual machines from one server to another without incurring application downtimes. This same flexibility is also desirable *across* geographically distributed data centers. Such cross-data-center management requires efficient migration of both memory and disk state between data centers, overcoming constraints imposed by the WAN connectivity between them. This would enable a range of new data center management techniques such as cloud bursting, where applications are shifted between sites to obtain additional resources on demand, and follow-the-sun provisioning, where an application travels the globe to position itself near its active users.

Although the abstraction of treating resources that span data centers and cloud providers as a single unified pool of resources is attractive, the reality of these resources being distributed across significant geographic distances and interconnected via static wide area network links (WANs) conspire to make the realization of this vision difficult. Several challenges need to be addressed to realize the above use-cases.

Minimize Downtime: Migration of application VMs and their data may involve copying tens of gigabytes of state or more. It is desirable to mask the latency of this data copying overhead by minimizing application downtimes during the migration. One possible solution is to support live migration of virtual machines over a WAN, where data copying is done in the background while the application continues to run, followed by a quick switchover to the new location. While live migration techniques over local area network (LAN) are well known, WAN

Manuscript received February 09, 2013; revised October 14, 2013 and February 16, 2014; accepted June 25, 2014; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. Liu. Date of publication August 19, 2014; date of current version October 13, 2015. This work was supported in part by the NSF under Grants CNS-0916972, CNS-0720616, CNS-0855128, and CNS-1253575 and AT&T under a VURI Award. This is an expanded version of the paper published in the Proceedings of the ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE), Newport Beach, CA, USA, March 9–11, 2011.

T. Wood, G. Liu, and L. Chaufournier are with The George Washington University, Washington, DC 20052 USA (e-mail: timwood@gwu.edu).

K. K. Ramakrishnan is with the University of California, Riverside, Riverside, CA 92521 USA.

P. Shenoy is with the University of Massachusetts Amherst, Amherst, MA 01003 USA.

J. Van der Merwe is with the University of Utah, Salt Lake City, UT 84112 USA.

J. Hwang is with the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2014.2343945

migration raises new challenges, such as the need to migrate disk state in addition to memory state.

Minimize Network Reconfigurations: Whereas VM migration over a LAN can be performed transparently from a network perspective (IP addresses remains unchanged, TCP connections move over, etc.), doing so transparently is a major challenge over a WAN. Different data centers and cloud sites support different IP address spaces, so additional network support is necessary if WAN migration is to remain transparent from a user and application perspective.

Handle WAN links: Migration of virtual machines over a LAN is relatively simple since data center LANs are provisioned using high-speed low-latency links. In contrast, WAN links interconnecting data centers of an enterprise and the connection to cloud sites may be bandwidth-constrained, and speed-of-light constraints dictate that interdata center latencies are significantly higher than in a LAN environment. Even when data centers are interconnected using well-provisioned links, it may not be possible to dedicate hundreds of megabits/second of bandwidth to a *single* VM transfer from one site to another. Furthermore, cloud sites charge for network usage based on the total network I/O from and to cloud servers. Consequently, WAN migration techniques must be designed to operate efficiently over low-bandwidth links and must optimize the data transfer volume to reduce the migration latency and cost.

In this paper, we propose a platform called *CloudNet* to achieve the vision of seamlessly connected resource pools that permit flexible placement and live migration of applications and their data across sites. CloudNet makes the following contributions:

1) the design and implementation of a cloud computing platform that seamlessly connects resources at multiple data center and enterprise sites;
2) a centralized virtual private network (VPN) Controller architecture that automates reconfiguration of VPN endpoints;
3) a holistic view of WAN VM migration that handles persistent storage, network connections, and memory state with minimal downtime;
4) optimizations that minimize total migration time, application downtime, and volume of data transferred;
5) a coordination system that synchronizes the migration of multitier applications.

We have implemented a prototype of Cloudnet using the Xen platform and a commercial layer-2 VPN implementation. We perform an extensive evaluation using three data centers spread across the continental US. Our results show CloudNet's optimizations decreasing memory migration and pause time by 30%–70% in typical link capacity scenarios; in a set of VM migrations over a distance of 1200 km, CloudNet saves 20 GB of bandwidth, a 50% reduction. We also evaluate application performance during migrations to show that CloudNet's optimizations reduce the window of decreased performance compared to existing techniques.

II. CloudNet Motivation and Overview

CloudNet provides a networking infrastructure for linking data centers to clouds and an optimized form of live VM migration. These two features are valuable for a variety of situations such as cloud bursting, enterprise IT consolidation, and "follow-the-sun" workloads. We have previously presented our vision for these use cases [39]; in this paper, we focus on the infrastructure required to meet these goals.

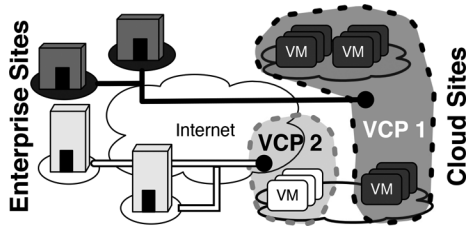

Fig. 1. Two VCPs isolate resources within the cloud sites and securely link them to the enterprise networks.

A. Resource Pooling: Virtual Cloud Pools

Enabling flexible, cross-data center management requires that: 1) data center and cloud sites can be seamlessly connected into pools; and that 2) applications can be efficiently transitioned between locations. CloudNet achieves this with the use of a Virtual Cloud Pool (VCP) abstraction that enables server resources across data centers and cloud providers to be logically grouped into a single server pool as shown in Fig. 1. The notion of a Virtual Cloud Pool is similar to that of a Virtual Private Cloud, which is used by Amazon's EC2 platform and was also proposed in our previous research [37]. Despite the similarity, the design motivations are different. In our case, we are concerned with grouping server pools across data centers, while Amazon's product seeks to provide the abstraction of a private cloud that is hosted on a public cloud. Both abstractions use VPNs as their underlying interconnection technology—we employ layer-2 VPNs to implement a form of network virtualization/transparency, while Amazon's VPC uses layer-3 VPNs to provide control over the network addressing of VM services.

The VCPs provided by CloudNet allow cloud resources to be connected to as securely and seamlessly as if they were contained within the enterprise itself. Furthermore, the cloud to enterprise mappings can be adjusted dynamically, allowing cross-data-center resource pools to grow and change depending on an enterprise's needs. With this flexible infrastructure in place, CloudNet seeks to enable efficient movement of applications between sites using optimized VM migration.

B. Efficient WAN Migration

Currently, moving an application to the cloud or another data center requires substantial downtime while application state is copied and networks are reconfigured. Alternatively, some applications can be easily replicated into the cloud while the original continues running. However, this only applies to a small class of applications (e.g., stateless Web servers or MapReduce style data processing jobs). These approaches are insufficient for the majority of enterprise applications, which have not been designed for ease of replication. Furthermore, many legacy applications can require significant reconfiguration to deal with the changed network configuration required by current approaches. In contrast, the live VM migration supported by CloudNet provides a much more attractive mechanism for moving between data centers because it is completely application-independent and can be done with minimal downtime.

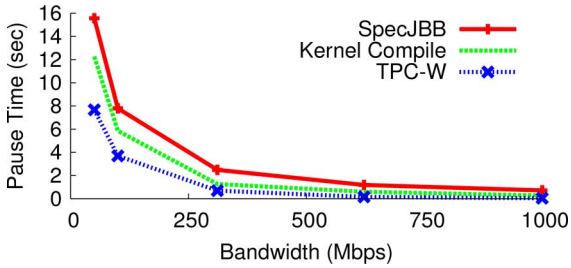

Fig. 2. Low-bandwidth links can significantly increase the downtime experienced during migration.

Most recent virtualization platforms support efficient migration of VMs within a local network [8], [27]. By virtue of presenting WAN resources as LAN resources, CloudNet's VCP abstraction allows these live migration mechanisms to function unmodified across data centers separated by a WAN. However, the lower bandwidth and higher latencies over WAN links result in poor migration performance. In fact, VMWare's preliminary support for WAN VM migration requires at least 622 Mb/s of bandwidth dedicated to the transfer and is designed for links with less than 5 ms latency [36]. Despite being interconnected using "fat" gigabit pipes, data centers will typically be unable to dedicate such high bandwidth for a *single* application transfer, and enterprises will want the ability to migrate a group of related VMs concurrently.

Fig. 2 shows the downtime of VMs running several different applications, as the available bandwidth is varied (assumes shared storage and a constant 10-ms round-trip latency). Note that performance decreases nonlinearly; migrating a VM running the SPECjbb benchmark on a gigabit link incurs a downtime of 0.04 s, but rises to 7.7 s on a 100-Mb/s connection. This nearly 200× increase is unacceptable for most applications and illustrates the importance of optimizing VM migration algorithms to better handle low-bandwidth connections.

In addition, current LAN-based VM migration techniques assume the presence of a shared file system that enables them to migrate only memory data and avoid moving disk state. A shared file system may not always be available across a WAN, or the performance of the application may suffer if it has to perform I/O over a WAN. Therefore, CloudNet coordinates the hypervisor's memory migration with a disk replication system so that the entire VM state can be transferred if needed.

Current LAN-based live migration techniques must be optimized for WAN environments, and cloud computing network infrastructure must be enhanced to support dynamic relocation of resources between cloud and enterprise sites; these challenges are the primary focus of this paper.

III. Network Support for VCPs

CloudNet implements the VCP abstraction to better integrate cloud and network services across multiple data centers. In this section, we describe how CloudNet manages the interface between the cloud and network, how it offers secure, seamless connectivity between data centers, and how it is able to dynamically reconfigure VCP topology based on changing enterprise needs.

A. Cloud and Network Management

CloudNet leverages existing virtualization technologies at the server, router, and network levels to create dynamic resource pools that can be transparently connected to enterprises. The CloudNet architecture is composed of two controllers that automate the management of resources in the provider network and in the cloud computing data centers.

The *Cloud Manager* is responsible for managing server and storage resources within cloud sites and enterprise data centers. In this paper, we focus on how the Cloud Manager's Migration Optimizer enables live migration across cloud sites. However, it could be further extended to automate these types of resource management decisions or offer additional services. While we consider the Cloud Manager as a single entity, it is actually composed of monitoring and control agents that are run in each cloud or enterprise data center used by a customer. The Cloud Manager utilizes several forms of virtualization so that physical resources can be multiplexed across many customers. In our current prototype, Xen is used to virtualize servers, and VLANs are used to partition the local area networks within each cloud data center. The Cloud Manager uses virtual (or "logical") routers to dynamically configure the customer edge (CE) routers associated with each VCP. Logical routers are a means to partition physical routers into slices, each with independent control planes. This means that full, physical routers do not need to be dedicated to each VCP and allows them to be created and reconfigured more rapidly.

The *Network Manager* is responsible for the creation and resource provisioning of VPNs. The Network Manager must be controlled by the network provider because it uses Multiprotocol Label Switching (MPLS) VPNs that span between the provider edge (PE) routers. In this paper we focus on how the Network Manager can dynamically modify the configurations of these PE routers in order to establish the new VPN topology required by a WAN migration.

Existing public cloud platforms such as Amazon EC2 provide much of the infrastructure needed to support the CloudNet platform. However, further communication between the cloud and network providers are necessary to determine the ID for the VLAN used by the Cloud Manager within its data center and connect it to the proper VPN endpoint (configured by the Network Manager). This could be achieved with minor changes to Amazon's Direct Connect feature that allows customers to extend VLANs between public and private sites. Of course, current public clouds do not allow customers to explicitly migrate VMs, but as these systems evolve, migration may become an important feature for customers looking to more easily deploy and shift applications between private and public infrastructure.

B. Seamless, Secure Cloud Connections

CloudNet uses MPLS-based VPNs to create the abstraction of a private network and address space shared by multiple data centers. Since addresses are specific to a VPN, the cloud operator can allow customers to use any IP address ranges that they prefer without concern for conflicts between cloud customers. CloudNet makes the level of abstraction even greater by using *virtual private LAN services* (VPLS) that bridge multiple MPLS endpoints onto a single LAN segment. This allows

cloud resources to appear *indistinguishable* from existing IT infrastructure already on the enterprise's own LAN. VPLS provides transparent, secure, and resource-guaranteed layer-2 connectivity without requiring sophisticated network configuration by end-users. This simplifies the network reconfiguration that must be performed when migrating VMs between data centers.

VPNs are already used by many large enterprises, and cloud sites can be easily added as new secure endpoints within these existing networks. VCPs use VPNs to provide secure communication channels via the creation of "virtually dedicated" paths in the provider network. The VPNs protect traffic between the edge routers at each enterprise and cloud site. Within a cloud site, the traffic for a given enterprise is restricted to a particular VLAN. This provides a secure end-to-end path from the enterprise to the cloud and eliminates the need to configure complex firewall rules between the cloud and the enterprise, as all sites can be connected via a private network inaccessible from the public Internet.

C. Dynamic VPN Connectivity to the Cloud

A straightforward implementation of VM migration between IP networks results in significant network management and configuration complexity [14]. As a result, *virtualizing network connectivity is key in CloudNet for achieving the task of WAN migration seamlessly relative to applications*. However, reconfiguring the VPNs that CloudNet uses to provide this abstraction has typically taken a long time because of manual (or nearly manual) provisioning and configuration. CloudNet explicitly recognizes the need to set up new VPN endpoints quickly and exploits the capability of BGP route servers [35] to achieve this.

In many cases, the destination data center will already be a part of the customer's virtual cloud pool because other VMs owned by the enterprise are already running there. However, if this is the first VM being moved to the site, then a new VPLS endpoint must be created to extend the VCP into the new data center.

Creating a new VPLS endpoint involves configuration changes on the data center routers, a process that can be readily automated on modern routers [20], [7]. The router must be reconfigured with a new virtual routing and forwarding (VRF) table to hold all of the routing information for the VPN. A traditional, but naive, approach would require modifying the router configurations at each site in the VCP so they all advertise and accept the proper *route targets*. A route target is an ID used to determine which VRF table processes a packet, which in turn determines how endpoints share connectivity. An alternative to adjusting the router configurations directly is to dynamically adjust the routes advertised by each site within the network itself. CloudNet takes this approach by having data center routers announce their routes to a centralized VPN Controller. The VPN Controller acts as an intelligent route server and is connected via BGP sessions to each of the cloud and enterprise data centers. The controller maintains a ruleset indicating which endpoints should have connectivity; as all route control messages pass through this VPN Controller, it is able to rewrite the route targets in these messages, which in turn control how the tunnels forming each VPLS are created.

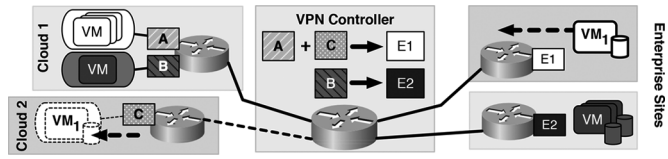

Fig. 3. VPN Controller remaps the route targets (A, B, C) advertised by each cloud data center to match the proper enterprise VPN (E1 or E2). To migrate VM_1 to Cloud Site 2, the VPN controller redefines E1's VPN to include route target A and C, then performs the disk and memory migration.

Fig. 3 illustrates an example where VM_1 is to be migrated from enterprise site E1 to Cloud Site 2. The VPN Controller must extend E1's VPLS to include route targets A and C, while Enterprise 2's VPLS only includes route target B. Once the change is made by the VPN Controller, it is propagated to the other endpoints via BGP. This ensures that each customer's resources are isolated within their own private network, providing CloudNet's virtual cloud pool abstraction. Likewise, the VPN Controller is able to set and distribute fine+grained access control rules via BGP using technologies such as Flowspec (RFC 5575).

Our approach allows for fast VCP reconfiguration since changes only need to be made at a central location and then propagated via BGP to all other sites. This provides simpler connectivity management compared to making changes individually at each site and allows a centralized management scheme that can set connectivity and access control rules for all sites.

Within the cloud site, the VRF for each customer is connected to a dedicated VLAN that isolates the enterprise's traffic within the data center. Note that CloudNet assumes that each customer *active* within a cloud site is given its own VLAN; if the 4096 VLAN limit is reached within a site, then data center scalability [24], [26] or MAC-in-MAC techniques (e.g., IEEE 802.1ad or 802.1Q-in-Q) may be explored.

IV. WAN VM Migration

Consider an organization that desires to move one or more applications (and possibly their data) between two data centers. Each application is assumed to be run in a VM, and we wish to live migrate those virtual machines across the WAN.

CloudNet uses these steps to live migrate each VM.

Step 1) Establish virtual connectivity between VCP endpoints.
Step 2) If storage is not shared, transfer all disk state.
Step 3) Transfer the memory state of the VM to a server in the destination data center as it continues running.
Step 4) Briefly pause the VM for the final transition of memory and processor state to the destination host. This process must not disrupt any active network connections between the application and its clients.

While these steps, illustrated in Fig. 4, are well understood in LAN environments, migration over the WAN poses new challenges. The constraints on bandwidth and the high latency found in WAN links makes Steps 2 and 3 more difficult since they involve large data transfers. The IP address space in Step 4 would typically be different when the VM moves between routers at

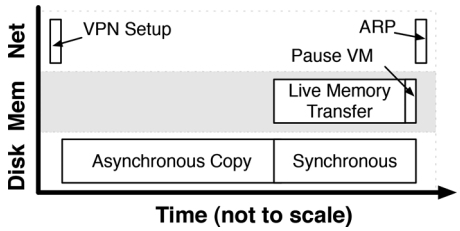

Fig. 4. Phases of a migration for nonshared disk, memory, and the network in CloudNet.

different sites. Potentially, application, system, router, and firewall configurations would need to be updated to reflect this change, making it difficult or impossible to seamlessly transfer active network connections. CloudNet avoids the reconfiguration issues in Step 1 by virtualizing the network connectivity as described in Section III and utilizes a set of migration optimizations to improve performance in the other steps.

A. Migrating Networks, Disk, and Memory

Here, we discuss the techniques used in CloudNet to transfer disk and memory and to maintain network connectivity throughout the migration. We discuss further optimizations to these approaches in Section IV-B.

1) Disk State Migration: LAN-based live migration assumes a shared file system holding VM disk images, eliminating the need to migrate disk state. As this may not be true in a WAN environment, CloudNet supports either shared disk state or a replicated system that allows storage to be migrated with the VM. If a VM relies on shared storage, then the network share must be securely accessible from both data centers. Otherwise, we have a "shared nothing" architecture where VM storage must be migrated along with the VM memory state.

CloudNet uses the DRBD disk replication system to migrate storage to the destination data center [10]. Once connectivity is established between sites, the replication system must copy the VM's disk to the remote host and must continue to synchronize the remote disk with any subsequent writes made at the primary. In order to reduce the performance impact of this synchronization, CloudNet uses DRBD's *asynchronous* replication mode during this stage. Once the remote disk has been brought to a consistent state, CloudNet switches to a *synchronous* replication scheme, and the live migration of the VM's memory state is initiated. During the VM migration, disk updates are synchronously propagated to the remote disk to ensure consistency when the memory transfer completes. When the migration completes, the new host's disk becomes the primary, and the origin's disk is disabled.

Migrating disk state typically represents the largest component of the overall migration time as the disk state may be in the tens or hundreds of gigabytes. Fortunately, disk replication can be enabled well in advance of a planned migration. Since the disk state for many applications changes only over very long timescales, this can allow the majority of the disk state to be transferred with relatively few wasted resources (e.g., network bandwidth). For unplanned migrations such as cloud bursting in response to a flash crowd, storage may need to be migrated on demand. CloudNet's use of asynchronous replication during bulk disk transfer minimizes the impact on application performance.

2) Transferring Memory State: Most VM migration techniques use a "pre-copy" mechanism to iteratively copy the memory contents of a live VM to the destination machine, with only the modified pages being sent during each iteration [8], [27]. At a certain point, the VM is paused to copy the final memory state. WAN migration can be accomplished by similar means, but the limited bandwidth and higher latencies can lead to decreased performance—particularly much higher VM downtimes—since the final iteration where the VM is paused can last much longer. CloudNet augments the existing migration code from the Xen virtualization platform with a set of optimizations that improve performance, as described in Section IV-B.

The amount of time required to transfer a VM's memory depends on its RAM allocation, working set size and write rate, and available bandwidth. These factors impact both the total time of the migration and the application-experienced downtime caused by pausing the VM during the final iteration. With WAN migration, it is desirable to minimize both these times as well as the bandwidth costs for transferring data. While pause time may have the most direct impact on application performance, the use of synchronous disk replication throughout the memory migration means that it is also important to minimize the total time to migrate memory state, particularly in high-latency environments.

Migrations over the WAN may also have a greater chance of being disrupted due to network failures between the source and destination hosts. The Cloud Manager can detect these failures and restart the migration or VPN reconfiguration. Because the switchover to the second site is performed only after the migration is complete, CloudNet will suffer no ill effects from this type of failure because the application will continue running on the origin site, unaffected. In contrast, some pull- or "post-copy"-based migration approaches start running the application at the destination prior to receiving all data, which could lead to the VM crashing if there is a network disconnection.

3) Maintaining Network Connections: Once disk and memory state have been migrated, CloudNet must ensure that the VM's active network connections are redirected. In LAN migration, this is achieved by having the destination host transmit an unsolicited ARP message that advertises the VM's MAC and IP address. This causes the local Ethernet switch to adjust the mapping for the VM's MAC address to its new switch port [8]. Over a WAN, this is not normally a feasible solution because the source and destination are not connected to the same switch. Fortunately, CloudNet's use of VPLS bridges the VLANs at each site, causing the ARP message to be forwarded over the Internet to update the Ethernet switch mappings at both sites. This allows open network connections to be seamlessly redirected.

In the Xen virtualization platform, this ARP is triggered by the VM itself after the migration has completed. In CloudNet, we optimize this procedure by having the destination host preemptively send the ARP message immediately after the VM is paused for the final iteration of the memory transfer. This can

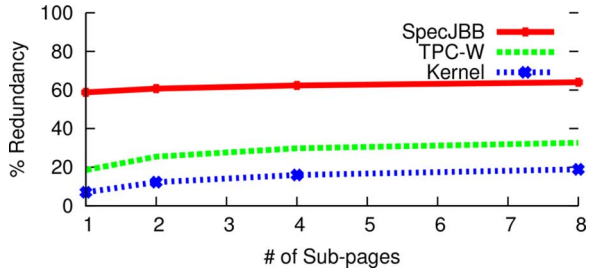

Fig. 5. Each application has a different level of redundancy. Using finer granularity finds more redundancy, but has diminishing returns.

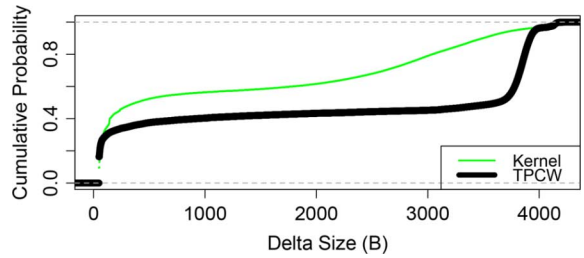

Fig. 6. During a kernel compile, many pages only experience very small modifications. TPC-W has some pages with small modifications, but more than half are almost completely changed.

reduce the downtime experienced by the VM by allowing the ARP to propagate through the network in parallel with the data sent during the final iteration. However, on our evaluation platform this does not appear to influence the downtime, although it could be useful with other router hardware since some implementations can cache MAC mappings rather than immediately updating them when an ARP arrives.

B. Optimizing WAN VM Migration

In this section, we propose a set of optimizations to improve the performance of VM migration over the WAN.

1) Content-Based Redundancy: Content-based redundancy (CBR) elimination techniques have been used to save bandwidth on links between network routers [2]. We use a similar approach to eliminate redundant data in VM memory and disk state. Disks can have large amounts of redundant data caused by either empty blocks or similar blocks of content across files. Likewise, a virtual machine can often have redundant pages in memory from similar applications or duplicated libraries.[1]

There are a variety of mechanisms that can be used to eliminate redundancy in a network transfer, and a good comparison of techniques is found in [1]. CloudNet can support any type of redundancy elimination algorithm; for efficiency, we use a block-based approach that detects identical, fixed-size regions in either a memory page or disk block. We have also tested a Rabin fingerprint-based redundancy elimination algorithm, but found it to be slower without substantially improving the redundancy detection rate.

CloudNet's block-based CBR approach splits each memory page or disk block into fixed-sized blocks and hashes their content. If a hash matches an entry in fixed-size, first-in–first-out (FIFO) caches maintained at the source and destination hosts, then a block with the same contents was sent previously. After verifying the pages match (in case of hash collisions), the migration algorithm can simply send the 32-bit cache entry index instead of the full block (e.g., 4 kB for a full memory page).

Fig. 5 shows the amount of memory redundancy found in several applications during migrations over a 100-Mb/s link as the number of blocks per page was varied. Increasing the number of blocks raises the level of redundancy that is found, but it can incur greater overhead since each block requires a hash table lookup. In CloudNet, we divide each page into four blocks since this provides a good tradeoff of detection rate versus overhead [40].

2) Using Page Deltas: After the first iteration, most of the pages transferred are ones that were sent previously, but have since been modified. Since an application may be modifying only portions of pages, another approach to reduce bandwidth consumption is to keep a cache of previously transmitted pages, and then only send the difference between the cached and current page if it is retransmitted. This technique was first demonstrated in high availability systems to reduce the bandwidth required for VM synchronization in a LAN [9], [12] and has since been considered for VM migrations [32], [38]. Support for delta compression has been added to recent versions of Xen (release 4.2), but the implementation is currently restricted to VM's running the Remus checkpointing system.

We have modified the Xen migration code so that if a page, or subpage block, does not match an entry in the cache using the CBR technique described previously, then the page address is used as a secondary index into the same cache. If the page was sent previously, then only the difference between the current version and the stored version of the page is sent.

Fig. 6 shows histograms of delta sizes calculated during migrations of two applications. A smaller delta means less data needs to be sent. Both applications have a large number of pages with only small modifications, but TPC-W also has a collection of pages that have been completely modified. This result suggests that page deltas can reduce the amount of data to be transferred by sending only the small updates, but that care must be taken to avoid sending deltas of pages that have been heavily modified.

3) Smart Stop and Copy: The Xen migration algorithm iterates until either a very small number of pages remain to be sent, it has already sent more than three times the VM's total memory, or a limit of 30 iterations is reached. At that point, the VM is paused, and all remaining pages are sent. While these rules are effective within a LAN, using these conditions tends to cause a large amount of data to be unnecessarily transferred in WAN settings.

Fig. 7 shows the number of pages remaining to be sent at the end of each iteration during a migration of a VM running a kernel compilation over a link with 622 Mb/s bandwidth and 5 ms latency. After the fourth iteration, there is no significant drop in the number of pages remaining. This indicates that: 1) a large number of iterations only increases total migration time

[1]Our implementation only supports removing memory redundancy during migrations, but we also use an offline CBR analysis tool to measure the potential for redundancy elimination in disks in Section VI-E.

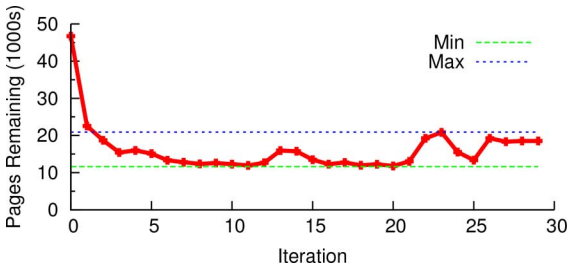

Fig. 7. Number of pages to be sent quickly levels off. Intelligently deciding when to stop a migration eliminates wasteful transfers and can lower pause time.

and data transfers; and 2) intelligently picking when to stop iterating could decrease both total and pause time. For the migration shown, picking the optimal point to stop the migration would reduce pause time by 40% compared to the worst stopping point.

CloudNet uses a *Smart Stop and Copy* optimization to reduce the number of unnecessary iterations and to pick a stopping point that minimizes pause time. Unfortunately, these two goals are potentially conflicting. Stopping after only a few iterations would reduce *total time*, but may result in a larger *pause time*, which can potentially have a larger impact on application performance. The Smart Stop algorithm is designed to balance this tradeoff by minimizing pause time without significantly increasing total time.

Smart Stop considers the migration as two phases: first where the majority of VM data is transferred, and the second where the remaining pages in a VM's working set are overwritten faster than they can be transferred. Smart Stop detects the transition between these phases by monitoring the number of dirtied and sent pages; if more pages are dirtied than sent in one iteration, then continuing to iterate may not provide a significant benefit.

While it is possible to stop the migration immediately at the point where as many pages are dirtied as sent, we have found that often the random fluctuations in how pages are written to can mean that waiting a few more iterations can result in a lower pause time with only a marginal increase in total time. Based on this observation, Smart Stop switches mode once it detects this crossover and begins to search for a local minimum in the number of pages remaining to be sent. If at the start of an iteration, the number of pages to be sent is less than any previous iteration in a sliding window, Smart Stop pauses the VM to prevent any more memory writes and sends the final iteration of memory data.

4) Synchronizing Arrivals: Consider a two-tier Web application with a front-end Web server and a back-end database. The database virtual machine may have a larger memory requirement than the Web server, which means that it may take substantially longer to perform its migration. This can result in a distributed application being split across the wide area network, which will seriously reduce performance since all inter-tier communication now must traverse high-latency links.

To prevent this problem, CloudNet supports *Synchronized Arrival* of virtual machines. This extends the Smart Stop algorithm described above so that if multiple, related virtual machines are moved simultaneously, the final migration iteration will not be performed until all VMs have transitioned to the phase where more data are dirtied than sent. As shown in Fig. 8, the migration process on each host has been modified to send a status message to CloudNet's Migration Coordinator. Rather than have each host decide when to finish a migration, the coordinator decides based on knowledge of all active related migrations.

We only apply this optimization to the memory portion of a migration—since CloudNet performs the bulk disk migration by asynchronously propagating writes, there is minimal application performance impact during this stage as we will show in Section VI-D. Therefore, synchronization is most important for the final stage of the memory migration.

5) Where to Optimize: CloudNet implements the above optimizations within the hypervisor itself. While it is possible to perform some WAN optimizations such as redundancy elimination in network middleboxes [2], the Page Delta optimization relies on memory page address information that can only be obtained from within the hypervisor. Furthermore, VM migrations are typically encrypted to prevent eavesdroppers, and network-level CBR generally does not work over encrypted streams [1].

We build our optimizations into Xen's Domain-0 as shown in Fig. 8. CloudNet uses individual caches for each virtual machine memory or disk migration, meaning that it cannot detect redundant data across multiple VMs. Previous work has demonstrated that different virtual machines often have some identical pages in memory, e.g., for common system libraries [3], [13], however our evaluation in Section VI-M illustrates that most redundancy can be accounted for within just a single VM's memory. Different disks, on the other hand, often contain substantial amounts of duplication [21], so using a site-wide disk cache could provide a benefit.

V. CloudNet Implementation

We have implemented a prototype of CloudNet that uses three key building blocks: 1) the Xen virtualization platform[2]; 2) the DRBD storage replication protocol; and 3) a commercial router-based VPLS/layer-2 VPN implementation. Our CloudNet prototype assumes that each data center runs the Xen virtualization platform on its servers and runs applications inside Xen virtual machines. Application data are assumed to be either stored on a storage area network (SAN) (in which case, it is assumed to be accessible across data centers and not migrated) or stored on disks that are local to each data center (in which case it must be migrated along with an application). In the latter case, we use the DRBD storage replication software to migrate data from one data center to another. Lastly, we assume that each data center employs routers that provide layer-2 VPN support with VPLS; our current prototype relies on Juniper's commercial implementation of VPLS.

A. VPLS Router Reconfiguration

CloudNet must be able to dynamically manipulate the routers at each data center site in order to create VPN endpoints. To do this, we use Juniper routers that have a remote API that can be used to programmatically adjust the router configuration. Each site in CloudNet has a physical router that is split up into virtual

[2]Our original implementation [38] used Xen 3.4.1, but we have now ported most of our changes to Xen 4.2.1, which is used in Sections VI-K and VI-J

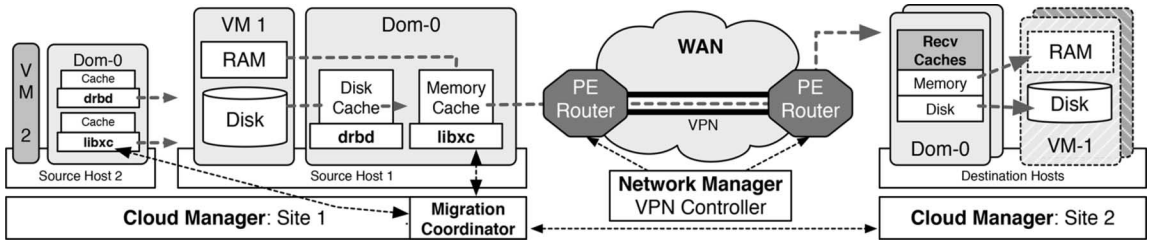

Fig. 8. After the VPN Controller establishes connectivity, the Cloud Manager's Migration Coordinator initiates each transfer. CloudNet uses separate caches for each memory and disk migration.

routers, one per VPLS endpoint. When a VM is migrated to a new site, CloudNet's Network Manager accesses the Juniper management interface to create the new logical router and to adjust the VPN Controller's mapping table.

B. Storage Migration With DRBD

DRBD is a storage replication system that has been integrated into the main line linux kernel [10]. CloudNet employs DRBD to migrate disk state of an application between data centers. The migration proceeds in two steps. First, a blank disk is created at the target. CloudNet then uses DRBD's asynchronous mode to perform an iterative precopy, transmitting all disk blocks to the destination and updating any subsequently modified blocks. Once the disks are synchronized, DRBD is switched to synchronous mode and initiates the memory state migration. This keeps both disks in sync by synchronously propagating all writes to both disks while Xen's memory state is migrated. Finally, DRBD is switched to dual primary mode during the switchover, allowing the VM to write to the disk at the destination host once the migration is complete. At this point, the source disk can be disconnected and its data deleted if appropriate.

Disk transfers can contain large amounts of redundant data. Our redundancy elimination code is not integrated with the DRBD synchronization protocol, however we are able to evaluate the potential benefit of this optimization by analyzing disk images with an offline CBR elimination tool.

C. Memory Optimizations

CloudNet extends Xen's live migration code as follows.

Content-Based Redundancy: CloudNet adds a content indexed cache that is checked before sending each page or portion of a page. Migration in Xen is performed by the *libxc* user space domain management tools. We have modified this library to generate fingerprints for each memory page using the Super Fast Hash Algorithm. We adjust the migration communication protocol so that page fingerprints can be sent instead of the full page, allowing the destination host to recover the data page from its own cache. Otherwise, the full page is sent, and the page and fingerprint are added to the cache at each end of the migration connection.

Page Deltas: To use page deltas, a second index into the page cache is created based on the page address. If the sender finds the page in the cache based on its address, then the current and cached pages are XOR'd to find the different bits. The XOR'd page is then run length encoded (RLE) to provide a basic form of compression. Since the RLE algorithm can potentially increase the size of the data sent if a page has been significantly modified, only pages with an RLE size less than a threshold are sent in their compressed form.

Smart Stop & Copy: We have adjusted the migration code to use Xen's dirty bitmap to calculate the number of pages remaining to be sent at the end of each iteration. These data are used to decide if the migration should continue through another iteration, or if it should be ended early. This change adds only a few dozen lines added to Xen's migration code.

Synchronized Arrival: We modify the migration code so that it connects to a coordination server running on the Cloud Manager when migrations begin. This server receives progress updates from each running migration including the current iteration, the number of memory pages sent, the number of pages dirtied, and the number of pages remaining, as well as an application ID for each VM. Since this communication is all within the LAN of the source data center, it incurs negligible overhead. If two VMs have the same ID, then the coordination server will apply the Smart Stop algorithm described above to both simultaneously, preventing one VM from finishing its migration significantly earlier than the other.

VI. Evaluation

This section evaluates the benefits of each of our optimizations and studies the performance of several different application types during migrations between data center sites under a variety of network conditions. We also study migration under three use-case scenarios: Section VI-D illustrates a cloud burst, Section VI-H studies multiple simultaneous migrations as part of a data center consolidation effort, and Section VI-I looks at the cost of disk synchronization in a follow-the-sun scenario.

A. Testbed Setup

We have evaluated our techniques between three data center sites spread across the US, and interconnected via an operational network, as well as on a laboratory testbed that uses a network emulator to mimic a WAN environment.

Data Center Prototype: We have deployed CloudNet across three data centers in Illinois, Texas, and California. Our prototype is run on top of the ShadowNet infrastructure that is used by CloudNet to configure a set of logical routers located at each site [6]. At each site, we have Sun servers with dual quad-core Xeon CPUs and 32 GB of RAM. We use Juniper M7i routers to create VPLS connectivity between all sites. We use the California site to run application clients, and migrate VMs between Texas and Illinois. Network characteristics between sites are variable since the data centers are connected via the Internet;

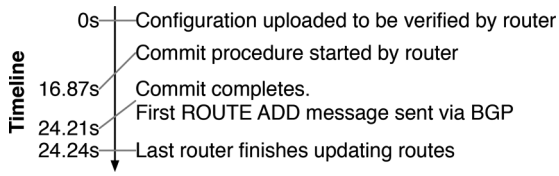

Fig. 9. Timeline of operations to add a new endpoint.

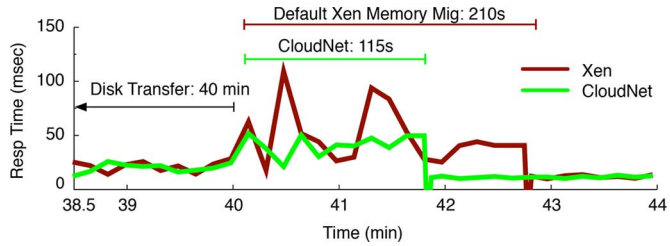

Fig. 10. Response times rise to an average of 52 ms during the memory migration, but CloudNet shortens this period of reduced performance by 45%. Response time drops to 10 ms once the VM reaches its destination and can be granted additional resources.

we measured an average round-trip latency of 27 ms and a max throughput of 465 Mb/s between the sites used for migrations.

Lab Testbed: Our lab testbed consists of multiple server/router pairs linked by a VPLS connection. The routers are connected through gigabit ethernet to a PacketSphere Network Emulator capable of adjusting the bandwidth, latency, and packet loss experienced on the link. We use this testbed to evaluate WAN migrations under a variety of controlled network conditions.

B. Applications and Workloads

Our evaluation studies three types of business applications. We run each application within a Xen VM and allow it to warm up for at least 20 min prior to migration.

SPECjbb 2005 is a Java server benchmark that emulates a client/server business application [30]. The majority of the computation performed is for the business logic performed at the application's middle tier. SPECjbb maintains all application data in memory, and only minimal disk activity is performed during the benchmark.

Kernel Compile represents a development workload. We compile the Linux 2.6.31 kernel along with all modules. This workload involves moderate disk reads and writes, and memory is mainly used by the page cache. In our simultaneous migration experiment, we run a compilation cluster using *distcc* to distribute compilation activities across several VMs that are all migrated together.

TPC-W is a Web benchmark that emulates an Amazon.com-like retail site [33]. We run TPC-W in a two-tier setup using Tomcat 5.5 and MySQL 5.0.45. Both tiers are run within a single VM. Additional servers are used to run the client workload generators, emulating 600 simultaneous users accessing the site using the "shopping" workload that performs a mix of read and write operations. The TPC-W benchmark allows us to analyze the client perceived application performance during the migration, as well as verify that active TCP sessions do not reset during the migration.

C. VPN Endpoint Manipulation

Before a migration can begin, the destination site may need to be added to the customer's VPN. This experiment measures the time required for CloudNet's VPN Controller to add the third data center site to our Internet-based prototype by manipulating route targets. Fig. 9 shows a timeline of the steps performed by the VPN Controller to reconfigure its intelligent route server. The controller sends a series of configuration commands followed by a commit operation to the router, taking a total of 24.21 s to be processed on our Juniper M7i routers; these steps are manufacturer-dependent and may vary depending on the hardware. As the intelligent route server does not function as a general-purpose router, it would be possible to further optimize this process if reduction in VPN reconfiguration time is required.

Once the new configuration has been applied to the router maintained by the VPN controller, the updated information must be propagated to the other routers in the network. The information is sent in parallel via BGP. On our network where three sites need to have their routes updated, the process completes in only 30 ms, which is just over one round-trip time (RTT). While propagating routes may take longer in larger networks, the initial intelligent route server configuration steps will still dominate the total cost of the operation.

D. Cloud Burst: Application Performance

Cloud Bursting allows an enterprise to offload computational jobs from its own data centers to the cloud. Current cloud bursting techniques require applications to be shut down in the local site and then restarted in the cloud; the live WAN migration supported by CloudNet allows seamless movement from an enterprise data center to the cloud.

We consider a cloud bursting scenario where a live TPC-W Web application must be moved from an overloaded data center in Illinois to one in Texas without disrupting its active clients; we migrate the VM to a more powerful server and increase its processor allocation from one to four cores once it arrives at the new data center location. In a real deployment, a single VM migration would not have access to the full capacity of the link between the data centers, so we limit the bandwidth available to 85 Mb/s; the VM is allocated 1.7 GB of RAM and has a 10-GB disk, similar to a "small" VM instance on Amazon EC2.[3] We assume that CloudNet has already configured the VPN endpoint in Texas as described in Section VI-C. After this completes, the DRBD subsystem begins the initial bulk transfer of the virtual machine disk using asynchronous replication; we discuss the disk migration performance details in Section VI-E and focus on the application performance during the memory migration here.

The full disk transfer period takes 40 min and is then followed by the memory migration. Fig. 10 shows how the response time of the TPC-W Web site is affected during the final 1.5 min of the storage transfer and during the subsequent memory migration

[3]Small EC2 instances have a single CPU, 1.7 GB RAM, a 10-GB root disk, plus an additional 150-GB disk. Transferring this larger disk would increase the storage migration time, but could typically be scheduled well in advance.

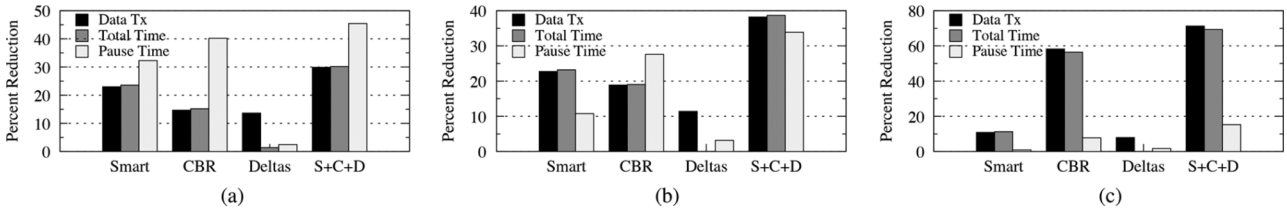

Fig. 11. CloudNet's optimizations affect different classes of application differently depending on the nature of their memory accesses. Combining all optimizations greatly reduces bandwidth consumption and time for all applications. (a) Kernel Compile. (b) TPC-W. (c) SPECjbb.

when using both default Xen and CloudNet with all optimizations enabled. During the disk transfer period, the asynchronous replication imposes negligible overhead; average response time is 22 ms compared to 20 ms prior to the transfer. During the VM migration itself, response times become highly variable, and the average rises 2.5× to 52 ms in the default Xen case. This overhead is primarily caused by the switch to synchronous disk replication—any Web request the involves a write to the database will see its response time increased by at least the round-trip latency (27 ms) incurred during the synchronous write. As a result, it is very important to minimize the length of time for the memory migration in order to reduce this period of lower performance. After the migration completes, the response time drops to an average of 10 ms in both cases due to the increased capacity available for the VM.

While both default Xen and CloudNet migrations do suffer a performance penalty during the migration, CloudNet's optimizations reduce the memory migration time from 210 to 115 s, a 45% reduction. CloudNet also lowers the downtime by half, from 2.2 to 1 s. Throughout the migration, CloudNet's memory and disk optimizations conserve bandwidth. Using a 100-MB cache, CloudNet reduces the memory state transfer from 2.2 to 1.5 GB. Furthermore, the seamless network connectivity provided by the CloudNet infrastructure prevents the need for any complicated network reconfiguration and allows the application to continue communicating with all connected clients throughout the migration. This is a significant improvement compared to current cloud bursting techniques that typically cause lengthy downtime as applications are shut down, replicated to the second site, and then rebooted in their new location.

E. Disk Synchronization

Storage migration can be the dominant cost during a migration in terms of both time and bandwidth consumption. During the initial bulk transfer phase, the DRBD system used by CloudNet transfers disk blocks to the destination by reading through the source disk at a constant rate (4 MB/s) and transmitting the nonempty blocks. Thus while the TPC-W application in the previous experiment was allocated a 10-GB disk, only 6.6 GB of data are transferred during the migration.

The amount of storage data sent during a migration can be further reduced by employing redundancy elimination on the disk blocks being transferred. Using a small 100-MB redundancy elimination cache can reduce the transfer to 4.9 GB, and a larger 1-GB cache can lower the bandwidth consumption to only 3.6 GB. Since the transfer rate is limited by the disk read speed, disk migration takes the same amount of time with and without CloudNet's optimizations. However, the use of content-based redundancy significantly reduces bandwidth costs during the transfer.

TABLE I
CloudNet Reduces Bandwidth, Total Time, and Pause Time During Migrations Over a 100-Mb/s Link With Shared Disk

	Data Tx (GB)	Total Time (s)	Pause Time (s)
TPC-W	1.5 → 0.9	135 → 78	3.7 → 2.3
Kernel	1.5 → 1.1	133 → 101	5.9 → 3.5
SPECjbb	1.2 → 0.4	112 → 35	7.8 → 6.5

F. Memory Transfer

To understand each memory optimization's contribution, we analyze migration performance using VMs allocated 1 GB of RAM running each of our three test applications; we create the VMs on a shared storage device and perform the migrations over a 100-Mb/s link with 20 ms RTT in our local testbed.

Fig. 11 shows each of CloudNet's optimizations enabled individually and in combination. We report the average improvement in total time, pause time, and data transferred over four repeated migrations for each optimization. Overall, the combination of all optimizations provides a 30%–70% reduction in the amount of data transferred and total migration time, plus up to a 50% reduction in pause time. Table I lists the absolute performance of migrations with the default Xen code and with CloudNet's optimizations.

Smart Stop: The Smart Stop optimization can reduce the data transferred and total time by over 20% (Fig. 11). Using Smart Stop lowers the number of iterations from 30 to an average of 9, 7, and 10 iterations for Kernel Compile, TPC-W, and SPECjbb, respectively. By eliminating the unnecessary iterations, Smart Stop saves bandwidth and time.

Smart Stop is most effective for applications with a large memory working set. In TPC-W, memory writes are spread across a database, and thus it sees a large benefit from the optimization. In contrast, SPECjbb repeatedly updates a smaller region, and these updates occur fast enough that the migration algorithm defers those pages until the final iteration. As a result, only a small number of pages would have been sent during the intermediate iterations that Smart Stop skips.

Fig. 12 shows the total number of pages sent in each iteration, as well as how much of the data is *final*—meaning it does not need to be retransmitted in a later iteration—during a TPC-W migration (we ignore the first iteration where the majority of the VM's memory is copied since more than 98% of that data is "final" and not retransmitted). After the second iteration, TPC-W sends over 20 MB per iteration, but only a small fraction of the total data sent is *final*—the rest is resent in later iterations when pages are modified again. In contrast, during the

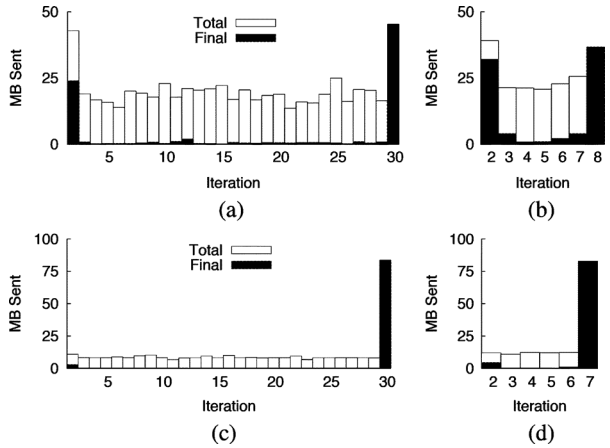

Fig. 12. Smart Stop reduces the iterations in a migration, significantly lowering the number of "useless" page transfers that otherwise are repeated. (a) TPC-W Default Xen. (b) TPC-W Smart. (c) SPECjbb Default Xen. (d) SPECjbb Smart.

SPECjbb migration shown in Fig. 12(c) and (d), Xen automatically detects that many pages are being constantly rewritten, and thus does not attempt to send them until the final iteration. In both cases, Smart Stop eliminates these unnecessary iterations to reduce the total data sent and migration time.

Smart Stop is also able to reduce the pause time of the kernel compile by over 30% [Fig. 11(a)]. This is because the compilation exhibits a variance in the rate at which memory is modified [Fig. 7(b)]. The algorithm is thus able to pick a better iteration to conclude the migration, minimizing the amount of data that needs to be sent in the final iteration.

Redundancy Elimination: Fig. 13(a) shows the amount of memory redundancy found in each application during migrations over a 100-Mb/s link when each memory page is split into four blocks and a 100-MB cache is used. SPECjbb exhibits the largest level of redundancy. However, the majority of the redundant data is from empty "zero" pages. In contrast, a kernel compilation has about 13% redundancy, of which less than half is zero pages. The CBR optimization eliminates this redundancy, providing substantial reductions in the total data transferred and migration time (Fig. 11). Since CBR can eliminate redundancy in portions of a page, it also can significantly lower the pause time since pages sent in the final iteration often have only small modifications, allowing the remainder of the page to match the CBR cache. This particularly helps the kernel compile and TPC-W migrations that see a 40% and 26% reduction in pause time, respectively. SPECjbb does not see a large pause time reduction because most of the redundancy in its memory is in unused zero pages, which are almost all transferred during the first iteration.

In Fig. 13(b) we show the amount of redundancy found as we vary the cache size. We have found that a cache in the 100–200-MB range is effective at detecting most redundancy. Even a very small 10-MB cache can remove a significant amount of redundancy, although this is primarily due to zero pages.

Page Deltas: The first iteration of a migration makes up a large portion of the total data sent since during this iteration the majority of a VM's memory—containing less frequently

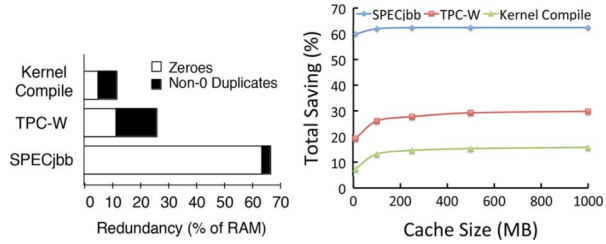

Fig. 13. (a) Each application has different levels of redundancy, in some cases mostly from empty zero pages. (b) Increasing the cache size leads to diminishing returns; we use a 100-MB cache in our remaining experiments.

TABLE II
PAGE DELTA OPTIMIZATION CANNOT BE USED DURING THE FIRST ITERATION, BUT IT PROVIDES SUBSTANTIAL SAVINGS DURING THE REMAINING ROUNDS

	Data Transferred (MB)		Page Delta
	Iter 1	Iters 2-30	Savings (MB)
TPC-W	954	315	172
Kernel	877	394	187
SPECjbb	932	163	127

touched pages—is transferred. The Page Delta optimization relies on detecting memory addresses that have already been sent, so it can only be used from the second iteration onward, and thus provides a smaller overall benefit, as seen in Fig. 11.

Table II shows the amount of memory data transferred during the first and remaining iterations during migrations of each application. While the majority of data is sent in the first round, during iterations 2–30, the Page Delta optimization still significantly reduces the amount of data that needs to be sent. For example, TPC-W sees a reduction from 487 to 315 MB, a 36% improvement.

Currently, the Page Delta optimization does not reduce migration time as much as it reduces data transferred due to inefficiencies in the code. With further optimization, the Page Delta technique could save both bandwidth and time.

Results Summary: The combination of all optimizations improves the migration performance more than any single technique. While the Page Delta technique only comes into effect after the first iteration, it can provide significant reductions in the amount of data sent during the remainder of the migration. The CBR-based approach, however, can substantially reduce the time of the first iteration during which many empty or mostly empty pages are transferred. Finally, Smart Stop eliminates many unnecessary iterations and combines with both the CBR and Page Delta techniques to minimize the pause time during the final iteration.

G. Impact of Network Conditions

We next use the network emulator testbed to evaluate the impact of latency and bandwidth on migration performance.

Bandwidth: Many data centers are now connected by gigabit links. However, this is shared by thousands of servers, so the bandwidth that can be dedicated to the migration of a single application is much lower. Here, we evaluate the impact of bandwidth on migrations when using a shared storage system. We vary the link bandwidth from 50 to 1000 Mb/s and use a constant 10-ms round-trip delay between sites.

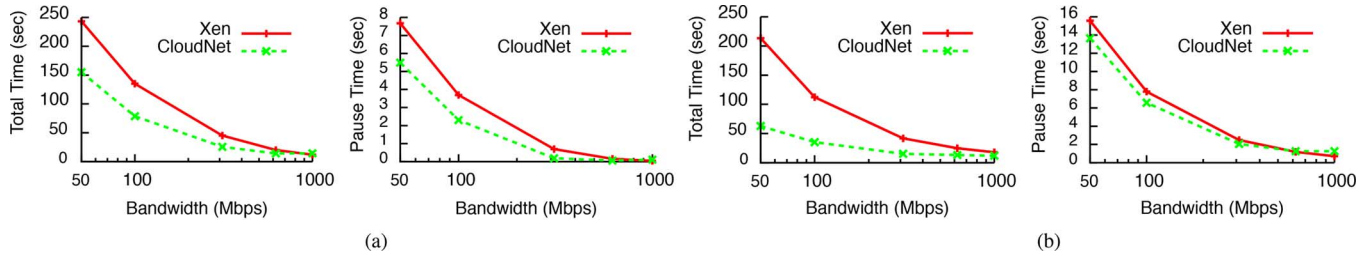

Fig. 14. Decreased bandwidth has a large impact on migration time, but CloudNet's optimizations reduce the effects in low bandwidth scenarios. (a) TPC-W. (b) SpecJBB.

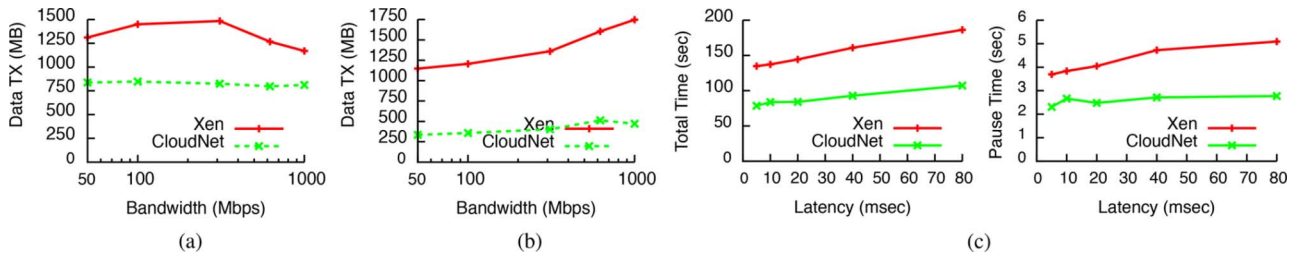

Fig. 15. (a) TPC-W bandwidth usage and (b) SPECjbb bandwidth usage. CloudNet's optimizations significantly reduce bandwidth consumption. (c) TPC-W latency impact. Increased latency has only a minor impact on the migration process, but may impact application performance due to synchronous disk replication.

Fig. 14 compares the performance of default Xen to CloudNet's optimized migration system. We present data for TPC-W and SPECjbb; the kernel compile performs similarly to TPC-W. Decreased bandwidth increases migration time for both applications, but our optimizations provide significant benefits, particularly in low-bandwidth scenarios. CloudNet also substantially reduces the amount of data that needs to be transferred during the migration because of redundancy elimination, page delta optimization, and the lower number of iterations, as seen in Fig. 15(a) and (b).

CloudNet's code presently does not operate at linespeed when the transfer rate is very high (e.g., about 1 Gb/s or higher *per VM transfer*). Thus, in high-bandwidth scenarios, CloudNet reduces the data transferred, but does not significantly affect the total migration or pause time compared to default Xen. We expect that further optimizing the CloudNet code will improve performance in these areas, allowing the optimizations to benefit even LAN migrations.

Latency: Latency between distant data centers is inevitable due to speed of light delays. This experiment tests how latency impacts migration performance as we adjust the delay introduced by the network emulator over a 100-Mb/s link. Even with TCP settings optimized for WAN environments, slow start causes performance to decrease some as latency rises. CloudNet's optimizations still provide a consistent improvement regardless of link latency as shown in Fig. 15(c). While latency has only a minor impact on total migration and pause time, it can degrade application performance due to the synchronous disk replication required during the VM migration. Fortunately, CloudNet's optimizations can significantly reduce this period of lowered performance.

H. Consolidation: Simultaneous Migrations

We next mimic an enterprise consolidation where four VMs running a distributed development environment must be transitioned from the data center in Texas to the data center in Illinois. Each of the VMs has a 10-GB disk (of which 6 GB is in use) and is allocated 1.7 GB of RAM and one CPU. The load on the cluster is created by repeatedly running a distributed kernel compilation across the four VMs. The maximum bandwidth available between the two sites was measured as 465 Mb/s with a 27-ms round-trip latency; note that bandwidth must be *shared* by the four simultaneous migrations.

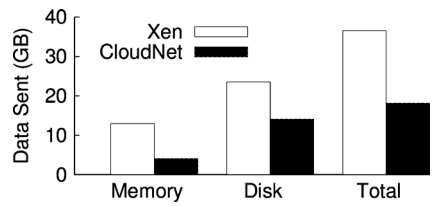

Fig. 16. CloudNet saves nearly 20 GB of bandwidth when simultaneously migrating four VMs.

We first run a baseline experiment using the default DRBD and Xen systems. During the disk synchronization period, a total of 24.1 GB of data is sent after skipping the empty disk blocks. The disk transfers take a total of 36 min. We then run the VM memory migrations using the default Xen code, which takes 245 s to complete all four migrations.

Next, we repeat this experiment using CloudNet's optimized migration code and a 1-GB CBR cache for the disk transfer. Our optimizations reduce the memory migration time—when application performance is impacted most—to only 87 s, and halves the average pause time from 6.1 to 3.1 s. Fig. 16 compares the bandwidth consumption of each approach. CloudNet reduces the data sent during the disk transfers by 10 GB and lowers the memory migrations from 13 to 4 GB. In total, the data transferred to move the memory and storage for all four VMs falls from 37.4 GB in the default Xen case to 18.5 GB when using CloudNet's optimizations.

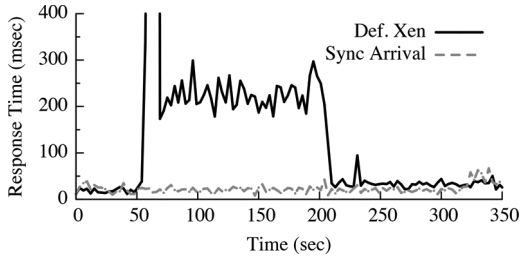

Fig. 17. If a multitier migration is not synchronized, performance can suffer badly.

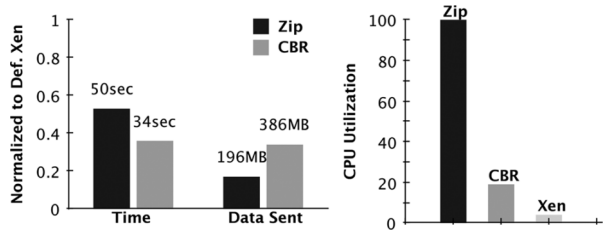

Fig. 18. Zip compression significantly reduces the data size, but has a very high computation cost.

I. Follow-the-Sun: Disk Synchronization

In a follow-the-sun scenario, one or more applications are moved between geographic locations in order to be co-located with the workforce currently using the application. In this experiment, we consider moving an application with a large amount of state back and forth between two locations. We focus on the disk migration cost and demonstrate the benefits of using incremental state updates when moving back to a location that already has a snapshot from the previous day.

We use the TPC-W Web application, but configure it with a much larger 45-GB database. The initial migration of this disk takes 3.6 h and transfers 51 GB of data to move the database and root operating system partitions. We then run a TCP-W workload that lasts for 12 h to represent a full workday at the site. After the workload finishes, we migrate the application back to its original site. In this case, only 723 MB of storage data need to be transferred since the snapshot from the previous day is used as a base image. This reduces the migration time to under 5 min, and the disk and memory migrations can be performed transparently while workers from either site are accessing the application. This illustrates that many applications with large state sizes typically only modify relatively small portions of their data over the course of a day. Using live migration and incremental snapshots allows applications to be seamlessly moved from site to site for relatively little cost and only minimal downtime.

J. Synchronized Migrations

To demonstrate the importance of synchronizing multitier migrations, we consider the case when TPC-W is split between a Tomcat VM with 0.5 GB of RAM and a MySQL VM with 4 GB of RAM. Both migrations are initiated at time $t = 0$, and we compare the response time seen by clients (not including network latency) with and without our synchronized arrival system.

As shown in Fig. 17, when no synchronization is used, the Tomcat VM finishes its migration after 59 s. This causes an immediate increase in response time, first since one VM is entirely paused, and then because suddenly all requests that involve a DB access must traverse the WAN. During this period, the average response time increases more than twentyfold, from 19 to 395 ms.

In contrast, Synchronized Arrival prevents the Tomcat migration from finishing prior to the database. As a result, performance is barely affected at all by the migration, with an average response time of 22 ms. In this case, the cost of migration in terms of data transferred does rise by 2.9 GB when using Synchronized Arrival. In Section VII, we describe future and related work that could reduce this cost.

K. Comparison to Zip Compression

Previous work has proposed using compression to reduce the amount of data that must be transferred during VM migrations [18]. Here, we evaluate the potential bandwidth savings and costs of such an approach. We migrate a VM with 1 GB of memory over an emulated link with 100 Mb/s bandwidth and 20 ms round-trip latency. The VM's workload is the kernel compile benchmark; we see similar behavior with the other applications.

Fig. 18 shows the data transfer, migration time, and CPU processing overhead of CloudNet with CBR and when migrating over a virtual IP tunnel configured to use zip compression. The results show that while the compressed tunnel provides the best reduction of data transferred (82% versus 65% with CBR), the reduction in migration time is not as great as when using CloudNet's CBR (47% versus 64% with CBR). The reason for this is shown in the CPU usage comparison—the compressed tunnel requires a significant amount of computation, essentially utilizing 100% of DomO's available CPU. This processing overhead slows down the data transfer compared to CBR, which incurs a relatively small computational cost (about 19% CPU utilization compared to 5% even with no optimizations). Thus, while the compressed tunnel lowers bandwidth consumption compared to CBR, its high computation cost actually increases the migration time. Particularly since migrations often occur during periods of peak demand, the high cost of zip compression may hurt performance without providing a significant benefit over CBR.

L. Comparison to Rabin Fingerprints

While all our results thus far have used block-based CBR, CloudNet could be easily extended to support Rabin fingerprints, another popular approach for eliminating redundancy in data streams. To compare these approaches, we record traces of all memory pages sent during migrations of the TPC-W and kernel compile VMs, without any of our optimizations, over a 100-Mb/s link with 10 ms latency. We then analyze this trace offline to compare the amount of redundancy detected with CloudNet's block-based CBR to the Rabin fingerprint-based approach implemented as described in [2].

Both approaches have a parameter that affects the granularity at which redundancy is found: the number of blocks that a page is divided into in the block-based approach, and the average

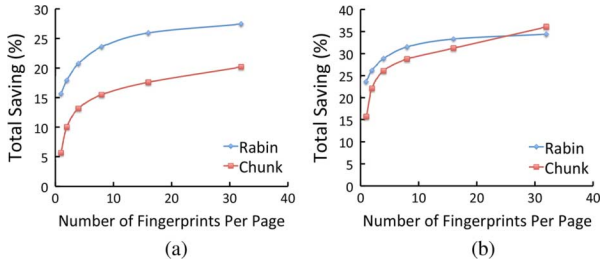

Fig. 19. Rabin fingerprinting techniques can detect finer grain redundancy at arbitrary offsets, but the benefit varies depending on the nature of the application. (a) Kernel Compile. (b) TPC-W.

number of fingerprints stored per page when using Rabin fingerprints. Fig. 19 shows how redundancy detection changes as we vary the size of blocks in CloudNet or the number of Rabin fingerprints stored per page. The benefit of Rabin fingerprints depends on the nature of the application—there is relatively little gain for TPC-W, particularly if pages are divided into four or more chunks. The kernel compile exhibits different behavior, with a clear gain from the Rabin approach. We speculate this may be because the kernel compile uses much of its memory for caching disk blocks, and many source code files may have similar content such as import statements or headers that may not be captured by the chunk-based approach if they occur at slightly different offsets in the file. However, our implementation of chunk-based redundancy elimination is more than an order of magnitude faster than the Rabin fingerprint-based approach. Since our goal is to provide low overhead optimizations, we focus on chunk-based CBR in this paper.

M. Cross VM Redundancy

CloudNet only seeks to find redundancy within the memory or disk data of the VM actively being migrated, however it is possible to use a joint cache that is shared across multiple VM migrations in the hopes of further eliminating redundant data. The benefit of such an approach when applied to disk data has been shown in prior work [21], but it has not been well studied in the context of virtual machine memory data.

To test the potential benefit of finding cross-VM duplicate pages, we measure the amount of redundancy that can be found when using one cache for multiple VM migrations. To provide a best-case scenario, we use a cache that is large enough to store all of the memory pages transmitted in traces from migrations of each of our three benchmark applications. We compare the amount of redundancy found when either a single joint cache or separate caches are used.

Surprisingly, we find that using a combined cache provides only a modest 8% improvement in the amount of redundancy that can be found. We believe this is because much of the redundant data found during a migration is self-contained within VMs [3]. This savings of about 120 MB out of the total 4.2 GB of data sent in the migrations represents memory regions that only appear at most once in a single VM, but also appear in different VMs. In contrast, the majority of redundancy found comes from regions that appear multiple times within the memory traces of a single VM, while zero pages are a notable candidate for this, there appear to be many other redundant pages as well. This is also explained in part by our use of subpage chunking to still find redundancy even if a memory page is slightly modified between migration rounds.

VII. Related Work

Private Clouds and Virtual Networks: The VIOLIN and Virtuoso projects use overlay networks to create private groups of VMs across multiple grid computing sites [29], [31]. VIOLIN also supports WAN migrations over well-provisioned links, but does not have a mechanism for migrating disk state. Overlay network approaches require additional software to be run on each host to create network tunnels. CloudNet places this responsibility on the routers at each site, reducing the configuration required on end hosts.

Our vision for Virtual Private Clouds was initially proposed in [37]. Subsequently, Amazon EC2 launched a new service also called "Virtual Private Clouds," which similarly uses VPNs to securely link enterprise and cloud resources. However, Amazon uses IPSec-based VPNs that operate at layer 3 by creating software tunnels between end hosts or IPSec routers. In contrast, CloudNet focuses on VPNs provided by a network operator. Network-based VPNs are typically realized and enabled by MPLS provider networks, following the "hose model" [11], and are commonly used by enterprises. Provider-based VPNs can be either layer-3 VPNs following RFC 2547, or layer-2 VPLS VPNs according to RFC 4761. CloudNet relies on network-based VPLS as it simplifies WAN migration, has lower overheads, and can provide additional functionality from the network provider, such as resource reservation.

LAN Migration: Live migration is essentially transparent to applications running inside the VM and is supported by most major virtualization platforms [8], [17], [27]. Work has been done to optimize migration within the LAN by exploiting fast interconnects that support remote memory access technology [16]. Breitgand *et al.* have developed a model-based approach to determine when to stop iterating during a memory migration [5], similar to Smart Stop. Their approach can allow them to more precisely predict the best time to stop, but it requires knowledge of the VM's memory behavior, and it is not clear how the model would perform if this behavior changes over time. CloudNet's CBR and Page Delta optimizations are simple forms of compression, and more advanced compression techniques such as those proposed by Jin *et al.* could provide further benefits in low-bandwidth WAN scenarios, although at the expense of increased CPU overhead [18]. The Remus project uses a constantly running version of Xen's live migration code to build an asynchronous high-availability system [9]. Remus obtains a large benefit from an optimization similar to CloudNet's Page Delta technique because it runs a form of continuous migration where pages see only small updates each iteration.

WAN Migration: VMware has support for WAN migration, but only under very constrained conditions: 622 MB/s link bandwidth and less than 5 ms network delay [36]. CloudNet seeks to lower these requirements so that WAN migration can become an efficient tool for dynamic provisioning of resources across data centers. Other companies such as F5 are developing WAN migration products that help simplify and optimize WAN migration. However, the details of these proprietary systems

have not been made public [25]. Past research investigating migration of VMs over the WAN has focused on either storage or network concerns. Bradford *et al.* describe a WAN migration system focusing on efficiently synchronizing disk state during the migration; they modify the Xen block driver to support storage migration and can throttle VM disk accesses if writes are occurring faster than what the network supports [4]. The VM Turntable Demonstrator showed a VM migration over intercontinental distances with latencies of nearly 200 ms; they utilize gigabit lightpath links and, like us, find that the increased latency has less impact on performance than bandwidth [34]. Harney *et al.* propose the use of Mobile IPv6 to reroute packets to the VM after it is moved to a new destination [15]; this provides the benefit of supporting layer-3 connections between the VM and clients, but the authors report a minimum downtime of several seconds due to the Mobile IP switchover, and the downtime increases further with network latency.

Shrinker uses content-based addressing to detect redundancy across *multiple* hosts at the destination site during VM migrations [28]. This could allow it to reduce bandwidth costs compared to CloudNet, but exposes it to security concerns due to hash collisions, although the likelihood of this can be bounded. A page-delta style optimization was employed by Svard *et al.* to reduce downtime during migrations; their approach is more streamlined than ours and provides a time benefit even in higher bandwidth settings [32]. Optimizations for storage migration have also been considered that use application workload characteristics to avoid resending frequently dirtied data blocks [23], [41]. While our implementation focuses on finding memory redundancy, Jin has shown that disks also include a large amount of redundant data [19].

CloudNet's Synchronized Arrival could be further optimized by using a model of migration completion time [5], [22] to determine the start time of each migration. This would prevent a fast migrating VM (i.e., one with a small memory size and/or working set) from needlessly iterating while waiting for a slower migration to finish. Furthermore, the speed of each migration could be controlled through dynamic bandwidth allocation schemes [8] to proactively make migrations complete at a time close to each other. Significant future work is needed in this area.

VIII. Conclusion

The scale of cloud computing is growing as business applications are increasingly being deployed across multiple global data centers. We have built CloudNet, a prototype cloud computing platform that coordinates with the underlying network provider to create seamless connectivity between enterprise and data center sites, as well as supporting live WAN migration of virtual machines. CloudNet supports a holistic view of WAN migration that handles persistent storage, network connections, and memory state with minimal downtime even in low-bandwidth, high-latency settings.

While existing migration techniques can wastefully send empty or redundant memory pages and disk blocks, CloudNet is optimized to minimize the amount of data transferred and lowers both total migration time and application-experienced downtime. Reducing this downtime is critical for preventing application disruptions during WAN migrations. CloudNet's use of both asynchronous and synchronous disk replication further minimizes the impact of WAN latency on application performance during migrations. We have demonstrated CloudNet's performance on both a prototype deployed across three data centers separated by over 1200 km and a local testbed. During simultaneous migrations of four VMs between operational data centers, CloudNet's optimizations reduced memory transfer time by 65%, and saved 20 GB in bandwidth for storage and memory migration. These improvements incur an overhead of less than 20% of one CPU core, a dramatic reduction compared to traditional compression techniques.

References

[1] B. Aggarwal *et al.*, "EndRE: An end-system redundancy elimination service for enterprises," in *Proc. NSDI*, 2010, p. 28.

[2] A. Anand, V. Sekar, and A. Akella, "SmartRE: An architecture for coordinated network-wide redundancy elimination," *Comput. Commun. Rev.*, vol. 39, no. 4, pp. 87–98, 2009.

[3] S. Barker, T. Wood, P. Shenoy, and R. Sitaraman, "An empirical study of memory sharing in virtual machines," in *Proc. USENIX Annu. Tech. Conf.*, Jun. 2012, p. 25.

[4] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg, "Live wide-area migration of virtual machines including local persistent state," in *Proc. 3rd ACM Int. Conf. Virtual Execution Environ.*, San Diego, CA, USA, 2007, pp. 169–179.

[5] D. Breitgand, G. Kutiel, and D. Raz, "Cost-aware live migration of services in the cloud," in *Proc. 3rd Annu. Haifa Exper. Syst. Conf., SYSTOR*, 2010, Art. no. 11.

[6] X. Chen, Z. M. Mao, and J. Van der Merwe, "ShadowNet: A platform for rapid and safe network evolution," in *Proc. USENIX Annual Technical Conference*, 2009, p. 3.

[7] Cisco, San Jose, CA, USA, "Cisco active network abstraction," [Online]. Available: http://www.cisco.com

[8] C. Clark *et al.*, "Live migration of virtual machines," in *Proc. NSDI*, May 2005, pp. 273–286.

[9] B. Cully *et al.*, "Remus: High availability via asynchronous virtual machine replication," in *Proc. NSDI*, 2008, pp. 161–174.

[10] LINBIT HA-Solutions GmbH, Vienna, Austria, "DRBD," [Online]. Available: http://www.drbd.org/

[11] N. G. Duffield *et al.*, "Resource management with hoses: Point-to-cloud services for virtual private networks," *IEEE/ACM Trans. Netw.*, vol. 10, no. 5, pp. 679–692, Oct. 2002.

[12] B. Gerofi, Z. Vass, and Y. Ishikawa, "Utilizing memory content similarity for improving the performance of replicated virtual machines," in *Proc. IEEE/ACM Conf. Utility Cloud Comput.*, 2011, pp. 73–80.

[13] D. Gupta *et al.*, "Difference engine: Harnessing memory redundancy in virtual machines," *Commun. ACM*, vol. 53, no. 10, pp. 85–93, 2010.

[14] M. Hajjat *et al.*, "Cloudward bound: Planning for beneficial migration of enterprise applications to the cloud," in *Proc. SIGCOMM*, 2010, pp. 243–254.

[15] E. Harney, S. Goasguen, J. Martin, M. Murphy, and M. Westall, "The efficacy of live virtual machine migrations over the internet," in *Proc. 3rd VTDC*, 2007, Art. no. 8.

[16] W. Huang, Q. Gao, J. Liu, and D. K. Panda, "High performance virtual machine migration with RDMA over modern interconnects," in *Proc. IEEE Int. Conf. Cluster Comput.*, 2007, pp. 11–20.

[17] Microsoft, Redmond, WA, USA, "Microsoft Hyper-V Server," [Online]. Available: http://www.microsoft.com/hyper-v-server

[18] H. Jin, L. Deng, S. Wu, X. Shi, and X. Pan, "Live virtual machine migration with adaptive memory compression," in *Proc. Cluster*, 2009, pp. 1–10.

[19] K. Jin and E. L. Miller, "The effectiveness of deduplication on virtual machine disk images," in *Proc. SYSTOR*, 2009, p. 7:17:12.

[20] Juniper Networks, Sunnyvale, CA, USA, "Configuration and diagnostic automation guide," [Online]. Available: http://www.juniper.net

[21] R. Koller and R. Rangaswami, "I/O deduplication: Utilizing content similarity to improve I/O performance," *Trans. Storage*, vol. 6, no. 13, pp. 1–26, Sep. 2010.

[22] H. Liu, H. Jin, C.-Z. Xu, and X. Liao, "Performance and energy modeling for live migration of virtual machines," *Cluster Comput.*, vol. 16, no. 2, pp. 249–264, Jun. 2013.

[23] A. Mashtizadeh, E. Celebi, T. Garfinkel, and M. Cai, "The design and evolution of live storage migration in VMware ESX," in *Proc. USENIX ATC*, Berkeley, CA, USA, 2011, p. 14.
[24] J. Mudigonda, P. Yalgandula, M. Al-Fares, and J. Mogul, "Spain: COTS data-center ethernet for multipathing over arbitrary topologies," in *Proc. NSDI*, 2010, p. 18.
[25] A. Murphy, "Enabling long distance live migration with F5 and VMware vMotion," Tech. rep., f5 Tech. Brief, 2011.
[26] R. N. Mysore *et al.*, "PortLand: A scalable fault-tolerant layer 2 data center network fabric," in *Proc. ACM SIGCOMM*, New York, NY, USA, 2009, pp. 39–50.
[27] M. Nelson, B.-H. Lim, and G. Hutchins, "Fast transparent migration for virtual machines," in *Proc. USENIX ATEC*, 2005, p. 25.
[28] P. Riteau, C. Morin, and T. Priol, "Shrinker: Improving live migration of virtual clusters over WANs with distributed data deduplication and content-based addressing," in *Proc. 17th Euro-Par*, 2011, vol. Part I, pp. 431–442.
[29] P. Ruth, J. Rhee, D. Xu, R. Kennell, and S. Goasguen, "Autonomic live adaptation of virtual computational environments in a multi-domain infrastructure," in *Proc. ICAC*, 2006, pp. 5–14.
[30] "The SPEC java server benchmark," 2013 [Online]. Available: http://spec.org/jbb2005/
[31] A. I. Sundararaj and P. A. Dinda, "Towards virtual networks for virtual machine grid computing," in *Proc. 3rd VM*, 2004, p. 14.
[32] P. Svärd, B. Hudzia, J. Tordsson, and E. Elmroth, "Evaluation of delta compression techniques for efficient live migration of large virtual machines," in *Proc. 7th ACM SIGPLAN/SIGOPS VEE*, New York, NY, USA, 2011, pp. 111–120.
[33] Transaction Processing Performance Council, "The TPC-W benchmark," 2013 [Online]. Available: http://www.tpc.org/tpcw/
[34] F. Travostino *et al.*, "Seamless live migration of virtual machines over the MAN/WAN," *Future Generation Comput. Syst.*, vol. 22, no. 8, pp. 901–907, Oct. 2006.
[35] J. Van der Merwe *et al.*, "Dynamic connectivity management with an intelligent route service control point," in *Proc. SIGCOMM Workshop Internet Netw. Manage.*, 2006, pp. 29–34.
[36] Cisco, San Jose, CA, USA, "Virtual machine mobility with VMware VMotion and Cisco data center interconnect technologies," Sep. 2009 [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns340/ns517/ns224/ns836/white_paper_c11-557822.pdf
[37] T. Wood, A. Gerber, K. Ramakrishnan, J. Van der Merwe, and P. Shenoy, "The case for enterprise ready virtual private clouds," in *Proc. USENIX HotCloud*, San Diego, CA, USA, Jun. 2009, Art. no. 4.
[38] T. Wood, K. K. Ramakrishnan, P. Shenoy, and J. Van der Merwe, "CloudNet: Dynamic pooling of cloud resources by live WAN migration of virtual machines," in *Proc. VEE*, Mar. 2011, p. 121132.
[39] T. Wood, K. K. Ramakrishnan, P. Shenoy, and J. Van der Merwe, "Enterprise-ready virtual cloud pools: Vision, opportunities, and challenges," *Oxford Comput. J.*, vol. 55, no. 8, pp. 995–1004, Jun. 2012.
[40] T. Wood *et al.*, "Memory buddies: Exploiting page sharing for smart colocation in virtualized data centers," in *Proc. ACM SIGPLAN/SIGOPS VEE*, Washington, DC, USA, Mar. 2009, pp. 31–40.
[41] J. Zheng, T. S. E. Ng, and K. Sripanidkulchai, "Workload-aware live storage migration for clouds," in *Proc. 7th ACM SIGPLAN/SIGOPS VEE*, New York, NY, USA, 2011, pp. 133–144.

Timothy Wood received the B.S. degree in electrical and computer engineering from Rutgers University, New Brunswick, NJ, USA, in 2005, and the M.S. and Ph.D. degrees in computer science from the University of Massachusetts, Amherst, MA, USA, in 2009 and 2011, respectively.

He is an Assistant Professor with the Computer Science Department, The George Washington University, Washington, DC, USA. His current research focuses on improving systems software support for data centers and cloud networks.

K. K. Ramakrishnan (S'76–A'83–M'03–SM'04–F'05) received the M.S. degree in automation from the Indian Institute of Science, Bangalore, India, in 1978, and the M.S. and Ph.D. degrees in computer science from the University of Maryland, College Park, USA, in 1981 and 1983, respectively.

He is a Professor with the Computer Science and Engineering Department, University of California, Riverside, CA, USA. From 1994 until 2013, he was with AT&T, most recently a Distinguished Member of Technical Staff with AT&T Labs—Research, Florham Park, NJ, USA. Prior to 1994, he was a Technical Director and Consulting Engineer in Networking with Digital Equipment Corporation, Littleton, MA, USA. Between 2000 and 2002, he was with TeraOptic Networks, Inc., Sunnyvale, CA, USA, as Founder and Vice President.

Dr. Ramakrishnan is an AT&T Fellow, recognized for his work on congestion control, traffic management and VPN services, and for fundamental contributions on communication networks with a lasting impact on AT&T and the industry.

Prashant Shenoy (M'99–SM'06–F'13) received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Bombay, India, in 1993, and the M.S. and Ph.D. degrees in computer science from the University of Texas, Austin, TX, USA, in 1994 and 1998, respectively.

He is currently a Professor of computer science with the University of Massachusetts, Amherst, MA, USA. His current research focuses on cloud computing and green computing.

Prof. Shenoy is a Distinguished Member of the ACM.

Jacobus Van der Merwe received the B. Eng and M. Eng degrees in electronic engineering from the University of Pretoria, Pretoria, South Africa, in 1989 and 1991, respectively, and the Ph.D. degree from the Computer Laboratory, Cambridge University, Cambridge, England, in 1998.

He is the Jay Lepreau Professor with the School of Computing and Director of the Flux Research Group with the University of Utah, Salt Lake City, UT, USA. He joined the University of Utah after 14 years with AT&T Labs—Research, Florham Park, NJ, USA. He does networking systems research in a broad range of areas including network management, control and operation, mobile networking, network evolution, network security, and cloud computing.

Jinho Hwang received the Ph.D. degree in computer science from The George Washington University, Washington, DC, USA, in 2013.

From 2005 to 2006, he worked as a Visiting Scholar with The George Washington University, and from 2007 to 2009, with the R&D center of POSCO ICT, Pohang, Korea. In the summers of 2012 and 2013, he worked with IBM Research and AT&T Labs—Research as a Research Intern, respectively. He is currently a Research Staff Member with the IBM T. J. Watson Research Center, Yorktown Heights, NY, USA. His area of interest is broadly in resource management and network virtualization of cloud computing.

Guyue Liu received the B.S. degree in electrical engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2012, and is currently pursuing the Ph.D. degree in computer science at The George Washington University, Washington, DC, USA.

Her research interests include software-defined networking and network virtualization.

Lucas Chaufournier is currently pursuing the Bachelor of Science degree in computer science at The George Washington University, Washington, DC, USA.

He is an Undergraduate Researcher with The George Washington University. His research interests include the intersection of systems and security as well as cloud computing.