A virtualization-based approach for zone migration in distributed virtual environments

Duong N. B. Ta, Thang Nguyen, Tran Nguyen, Nguyen Do, Xueyan Tang, Wentong Cai
School of Computer Engineering
Nanyang Technological University, Singapore
binhduong@ntu.edu.sg

Suiping Zhou
School of Computing
Teesside University
Middlesbrough, UK
s.zhou@tees.ac.uk

ABSTRACT

This paper deals with the zone migration problem in large-scale distributed virtual environments (DVEs), e.g., massively multi-player online games, distributed military simulations, etc. To support real-time interactions among thousands of concurrent, geographically separated clients, a distributed server architecture is generally needed. In such architecture, the large virtual world can be partitioned into multiple smaller zones, enabling load distributions or zone-to-server mappings to improve interactivity. For example, a zone might be mapped (assigned) to a server location near most of its clients to reduce network latency. In this paper, we consider the problem of live zone migration over wide area networks (WANs) to support DVE zone re-mapping or load re-distribution in a geographically distributed server infrastructure. We propose a virtualization-based zone migration approach, and develop several migration algorithms to effectively migrate multiple DVE zones.

We have implemented the proposed migration approach and algorithms in a tool for DVE performance enhancement and monitoring. Extensive experiments have also been conducted with an online multi-player game prototype constructed using the Torque 3D game engine. The results demonstrate the feasibility of our migration approach.

Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: Performance evaluation; C.2.4 [**Distributed Systems**]: Distributed applications

Keywords

Virtualization, distributed virtual environment, zone migration

1. INTRODUCTION

Distributed Virtual Environments (DVEs) are distributed systems that allow multiple geographically distributed clients (users) to explore and interact with each other in real time within a shared, computer-generated virtual world, in which each client is represented by an *avatar*. A client controls the behavior of his/her avatar by various *inputs*, and the *updates* of an avatar's state need to be sent to other clients in the same *zone* of the virtual world to support the interactions among clients. Examples of DVEs can be seen in multiple areas, such as collaborative design, military simulations, e-learning, virtual shopping mall, and multiplayer online games [14].

Typically, in large-scale DVEs with thousands of clients interacting simultaneously, the resource requirements in terms of network bandwidth, CPU cycles, memory, etc. are huge, and will increase very quickly as the number of concurrent clients increases. A distributed server infrastructure is usually required [13, 18] for such resource-intensive applications. In this architecture, each client connects to one of multiple geographically distributed servers in the system, and clients interact with each other through these servers. For load distribution, the large virtual world is spatially partitioned into several distinct *zones*, with each zone managed by only one server. A client only interacts with other clients in the same zone, and may move from one zone to another over time. As a server only needs to handle a few zones instead of the entire virtual world, the system becomes more scalable.

In such architecture and virtual world partitioning approach, it is desirable to migrate DVE zones across servers to ensure the workload is equally distributed [11, 17]; or to bring the zones closer to their respective clients [10, 16, 18]. The latter problem is usually referred to as the zone mapping problem, which arises due to the Internet's heterogeneity; and the fact that clients in a DVE are usually geographically separated. So, it is likely that a large number of clients in a zone may be far away (in terms of round-trip network latency) to the server hosting that zone, thus the DVE interactivity for these clients may be greatly degraded. Hence, there is a strong need for mechanisms to assign (map) DVE zones to servers in such a way that reduces the network latency between clients and servers. On the other hand, the former problem usually concerns how to assign zones to servers in DVEs with the objective of balancing the workload among servers [11, 17]. Furthermore, due to DVE dynamics, e.g., users joining and leaving, zone re-mapping or load re-distribution would be needed to maintain a certain level of performance. These important problems require the capability of flexible, seamless zone migration across servers in WANs. We should note that the migration must be live so

DISIO 2011, March 21, Barcelona, Spain

DOI 10.4108/icst.simutools.2011.245557

that the clients would be able to continue with the interaction in the virtual world during the migration.

In this paper, we propose a virtualization-based approach to deal with the live zone migration problem in DVEs. Essentially, we propose to encapsulate each distinct DVE zone in a virtual machine. When a zone needs to be moved to another physical server, the virtual machine hosting it will get migrated. Compared to traditional methods such as process migration [12], this approach inherits a number of major advantages of virtualization technology. For example, our approach is platform-independent, e.g., it can be used for any existing zone-based, multi-server DVEs without code modification/customization.

Below, we summarize the key contributions of this paper.

- We propose a virtualization-based, live zone migration approach over WANs for DVEs. For more flexibility, we consider the possibility of separating the migration of the virtual machine from the storage of each zone. We also develop two algorithms, namely sequential and parallel migration, for migrating multiple zones efficiently.
- We integrate the proposed zone mapping approach into a scalable and extensible software framework named DINE (DVE INteractivity Evaluation). DINE has been designed to support the development, integration, and performance evaluation of algorithms/methods to improve the interactive performance of large-scale DVEs. The framework is flexible enough to serve as either an evaluation platform for the development of new DVE interactivity enhancement methods, or a real-world performance monitoring and management suite for existing DVEs.
- We conduct extensive experiments to evaluate the effectiveness of the newly proposed approach. In particular, we have developed a DVE prototype with multiple zones using the Torque 3D game engine. We use this game to evaluate the performance of our proposed migration approach with multiple real and simulated, automated game clients.

The rest of the paper is organized as follows. Section 2 describes the zone migration problem. The proposed vitualization based approach and migration algorithms are presented in Section 3. Section 4 describes the implementation of the proposed approach. Simulation methodology and results are described in Sections 5 and 6, respectively. Section 7 concludes the paper.

2. ZONE MIGRATION IN DVES

Important problems in zone-based, large-scale DVEs with a geographically distributed server architecture such as zone mapping [16] require the capability of flexible zone migration across servers over WANs. Due to the Internet latency's fluctuations, clients' preferences and distributions, differences in time zones, etc., it is often necessary to quickly migrate many zones to and from geographically distributed server locations at the same time in order to achieve a desirable level of interactivity. Previous work, e.g., [6] have focused on migrating only a single game server in local area networks (LANs) where latency is minimal and bandwidth is abundant.

In this paper, we consider the problem of migrating multiple DVE zones in a geographically distributed server architecture. We assume that in such architecture, there are well-provisioned network links interconnecting the distributed servers (Figure 1). The whole virtual world is partitioned into a number of distinct zones, with each zone managed by only one server. Due to the triggering of zone mapping algorithms [16], or load distribution mechanisms [11], multiple zones might need to be migrated to different servers, as illustrated in Figure 1.

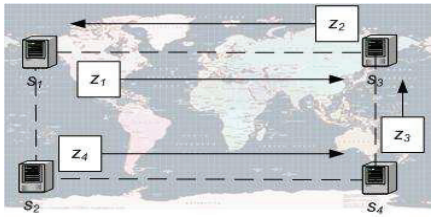


Figure 1: Zone migrations in a geographically distributed server architecture

Due to the highly interactive and human-in-the-loop nature of most DVEs, it is desirable that the zone migration will not affect the clients' experience of exploring and interacting in the virtual world. Therefore, two of the key factors that we consider in such migration are the total migration time, and the migration overhead. Since zone migration is needed to improve interactivity via load re-distribution or zone re-mapping, the migration time should be as short as possible so that DVE clients would enjoy a better level of interactivity sooner.

On the other hand, zone migration would involve transferring DVE contents, client network connections, etc. from one server location to another, possibly far away, server location. Such operations might consume a significant amount of server and network resources, which in turn may affect the clients' interactivity during migration. Hence, we need to carefully quantify and assess the effect of such migration overhead.

3. A VIRTUALIZATION-BASED APPROACH FOR DVE ZONE MIGRATION

Recently, server virtualization has been gaining popularity, and has become a key technology for server consolidation, e.g., lowering the number of physical servers in data centers while increasing server utilization; server/application isolation and security; fault tolerance; and supporting different platforms or legacy applications on the same set of hardware, etc. Essentially, virtualization hides the physical characteristics of computing infrastructures from the users; providing an abstract computing platform on which multiple virtual machines (VMs) can be run. The software suite that provides the abstract platform and controls the VMs is referred to as the *hypervisor*, or virtual machine monitor (VMM).

One of the key benefits of virtualization technology is the capability of live virtual machine migration across physical servers. Previous work has shown that migrating an entire VM (which includes its operating system and all applications) would help to avoid many problems introduced by migrating individual processes [6]. Most notably is the problem of "residual dependencies", in which the original server must still remain available and accessible over the network after the processes in question have been migrated. This is to make it possible to service certain system calls; or memory accesses on behalf of migrated processes.

Another benefit of VM migration over process migration is that in-memory state can be transferred in a consistent manner [6]. This means, for example, that we can migrate an online game server without asking clients to reconnect; which enables transparent migrations. Last but not least, VM migration offers the capability of migrating an entire set of applications to other physical machines without concerns about hardware compatibility.

In this paper, we propose an approach based on virtualization for DVE zone migration. Due to the geographically distributed server architecture, the zones will need to be able to migrate over WANs. In this approach, each DVE zone is encapsulated in a VM. Each VM has only one zone; and a physical server may handle multiple VMs (zones) as long as the server capacity permits. When a zone needs to be migrated to another physical server, the entire VM will get migrated. This virtualization-based approach is very flexible, as it can be used for any zone-based, multi-server DVEs without any modifications to the DVE's code.

We consider the problem of zone migration in DVEs, leveraging existing work in VM migration over WANs, e.g., [8, 20]. Being a class of human-in-the-loop and highly interactive applications, DVEs are more demanding in terms of CPU, memory, GPU and network resources; rather than being data-intensive like those applications considered in [8, 20] for instance. Therefore, we focus on the key issues about zone migration that might affect DVE's interactivity in a geographically distributed server architecture, namely the problem of storage's location, and migration algorithms.

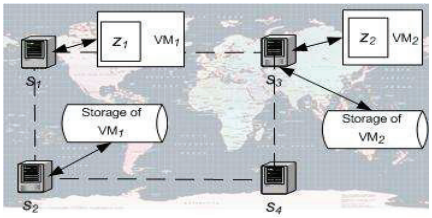


Figure 2: Virtualization-based approach for zone migration in DVEs

3.1 Storage's location

Usually, in a LAN environment, there will be centralized storage facilities to store the VM disk images. When a VM needs to be migrated to another physical server, only the operating system and its applications will be transferred. The disk image of that VM will remain where it is. Due to LAN's high bandwidth and minimal latency, such a remote storage approach might not be a serious problem for most of the applications running in the VM. On the other hand, it is not possible to eliminate WAN's latency; and migrating VM's storage over WANs may incur considerable overhead. Furthermore, DVEs require a high level of interactivity. Therefore, we need to carefully consider whether the VM's storage should be migrated together with the VM itself when the DVE zone gets migrated.

Figure 2 illustrates our virtualization-based approach for DVE zone migration. In our approach, we consider two separate components for each zone, namely the VM (the operating system and all its applications) and the storage for that VM. A VM's storage may reside at any physical server in the system, as long as its location does not affect the zone's interactivity significantly. When a zone needs to be migrated, the VM will be transferred first. Depending on the users' requirements, the VM's storage might get migrated to the same new physical server of the VM, or may remain at its current location. In Figure 2, the storage of zone z_1's VM is at server s_2, while its VM runs at server s_1. z_1's VM will need to access the storage via the inter-server network link between s_1 and s_2. On the other hand, zone z_2's VM and its storage are all at server s_3.

In such migration approach, we will have multiple VMs migrating across distributed physical servers freely, each of them maintaining an open network connection back to its storage residing somewhere in the system. We believe that this remote storage approach would not affect the DVE interactive performance significantly in most situations, as DVEs are not data-intensive applications. However, we will still need to carefully quantify and assess any possible effect that it might have in real-world scenarios.

3.2 Migration algorithms

Since a new zone mapping arrangement may require the migration of multiple zones across physical servers [16], we need to consider how to efficiently migrate a large number of zones. In this paper, we propose two algorithms to deal with such scenario, namely, parallel and sequential migration. Note that these algorithms can be used for both migration of VMs and storage.

3.2.1 Parallel migration

In the parallel migration algorithm, all the zone migrations will start at the same time. For example, in Figure 1, all the four zones z_1, z_2, z_3 and z_4 will be migrated concurrently. A parallel migration is expected to have short completion time. However, it may cause significant overhead on some physical servers in the system, thus possibly affecting the migration time and the interactivity performance of the clients in those servers. In Figure 1, a parallel migration may overload server s_3, as this server will need to handle the migrations of 3 zones at the same time: two incoming zones (z_1 and z_3) and one outgoing zone (z_2).

3.2.2 Sequential migration

In the sequential migration algorithm, at any point in time, there will be only one migration transaction for each physical server in the system. For example, a sequential migration for the scenario in Figure 1 might start with zone z_1's migration from s_1 to s_3. At the same time, zone z_4 can also be migrated from s_2 to s_4. However, z_3's migration will need to wait until these two migrations complete. Although sequential migration reduces concurrency, it might reduce the overhead on the physical servers. This in turn might help to shorten the overall migration time, and may provide a better level of interactivity for DVE clients during migration.

3.3 Related work

To the best of our knowledge, there is no existing work that directly addresses the problem of DVE zone migration in a geographically distributed server architecture, where physical servers are interconnected by links with bandwidth much lower and latency much higher compared to those in

LANs. Previous work such as [6] only considered a single DVE zone migration in LAN.

Recent research efforts have considered Internet/WAN VM migration [5, 7, 8, 19, 20]. For example, [7] uses mobile IPv6 to enable constant network connectivity during live migration of VMs over Internet. CloudNet [20] provides similar capabilities using Multi-Protocol Label Switching (MPLS) based VPNs, as well as a disk replication system for storage migration. [5] combines a block-level solution with precopying and write throttling to migrate an entire running web server and its storage, with minimal disruptions. [8] proposes a distributed storage access mechanism that supports live VM migration over WAN. It works as a storage server for a block-level storage I/O protocol such as iSCSI or NBD (Network Block Device). Most of the existing work focuses on maintaining seamless network connectivity during and after the process, and storage migration for data-intensive applications. Our work instead focuses on important issues in a virtualization-based approach for DVE zone migration, such as storage's location and how to migrate multiple zones efficiently. These problems might affect the interactive performance of DVEs significantly.

In [9], the authors have proposed an online game resource provisioning model using smaller and less expensive sets of self-owned data centers, complemented by virtualized cloud computing resources during peak hours. They have studied the impact of provisioning virtualized cloud resources, analyzed the components of virtualization overhead, and compared provisioning of virtualized resources with direct provisioning of data center resources. However, the impact of VM migrations on online games' performance has not been experimentally studied.

In [4], the authors have considered a combination of load distribution and increased resource locality by migrating online game state to an optimal location considering all users that are currently interacting in the game. The level of game-state granularity can range from entire virtual regions to single game objects. They have also highlighted that not all states related to an object need to be migrated, so some migration overhead can be reduced. To enable continuous interaction with the virtual environment during migration, remote method invocations with a distributed name service have been implemented. This approach might provide a lighter-weight migration facility compared to VM migration. However, it would be difficult to support existing DVEs. Consistency maintenance [21] might also be hard to implement. In our approach, the entire DVE zone's state is encapsulated in a single VM, thus consistency policies can be enforced relatively easy.

4. DESIGN AND IMPLEMENTATION

Figure 3 shows our implemented zone migration facility. We use Xen [3] version 3.4.1 as the VMM, and xNBD [8] as the storage back-end manager, all running on Linux kernel 2.6.31. We have designed the software to be flexible and extensible, for example, we can easily incorporate other VMMs such as KVM [2], or a different storage manager such as Distributed Replicated Block Device (DRBD) [1] if needed.

Xen [3] is a popular VMM for IA-32, x86-64, Itanium and ARM architectures. It allows several guest operating systems to run on the same computer hardware concurrently. Initial versions of Xen support only paravirtualization, which requires modifications of guest operating systems' kernels. Starting with Xen 3.0, hardware-assisted virtualization is supported, enabling unmodified guest operating systems to run within Xen VMs. This allows the virtualization of proprietary operating systems (for example Microsoft Windows). In this paper, we implement each DVE zone as a Windows XP VM.

Xen supports live migration of VMs between networked physical hosts with minimal disruption. Once a live migration starts, Xen VMM iteratively transfers the memory of the migrating VM to the destination machine without stopping the VM's execution. In the last iteration, the migrating VM needs to be stopped to carry out some synchronization before the VM continues its execution at the new destination. We use this feature to implement DVE zone migration.

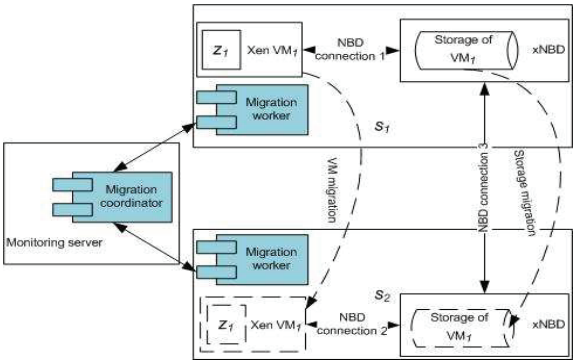


Figure 3: Implementation of the virtualization-based migration approach

Since DVE zones will need to be migrated over WANs, both the VM and its storage might need to be migrated. To enable storage migration, we use xNBD [8], which is a NBD (Network Block Device) server program. xNBD is fully compatible with the NBD client driver of the Linux kernel.

In Figure 3, each zone is encapsulated in a Xen VM. The storage of each VM is managed by xNBD. The Xen VM connects to its storage (regardless of the storage's location) using the standard NBD client program, which is included in all recent versions of the official Linux kernel. The key components in our implementation are the migration coordinator and the migration worker. Both of them are multi-threaded programs, and have been implemented in Java for better portability.

- **Migration coordinator:** This component runs on a centralized monitoring server, and manages the entire migration process of multiple zones. It receives a list of zones that need to be migrated, and schedules the migrations of those zones in a parallel or sequential manner. It sends control commands to and waits for completion signals from the migration workers.
- **Migration worker:** This component runs on each physical server in the DVE's infrastructure. Its main task is to carry out the migration for the zones hosted by its physical server. It receives commands from the migration coordinator regarding which zones, where and when they need to be migrated. Once a zone migration is completed, it will notify the coordinator. The migration worker communicates directly with Xen VMM and the xNBD via our own Linux shell scripts to facilitate VM and storage migration.

4.1 Integration into DINE

To examine the performance of the newly proposed zone migration approach under realistic scenarios, we have integrated it into our own scalable and extensible software framework named DINE (DVE INteractivity Evaluation) [15]. Essentially, DINE has been designed and implemented to support the development, integration, and performance evaluation of interactivity enhancement methods for large-scale DVEs. The framework is flexible enough to serve as either an evaluation platform for the development of new DVE interactivity enhancement methods, or a real-world performance monitoring and management suite for existing DVEs.

In this paper, we use the zone mapping algorithms [16] as a case study to demonstrate the capabilities of the newly proposed zone migration approach. We have implemented a complete solution starting from monitoring DVE interactive performance via scalable network latency measurement, triggering the zone mapping algorithms when the Quality of Service (QoS) level drops below a given threshold, and migrating some zones to appropriate servers to achieve better QoS according to the results of the zone mapping algorithms.

5. EVALUATION METHODOLOGY

5.1 The DVE prototype

Figure 4: The DVE prototype

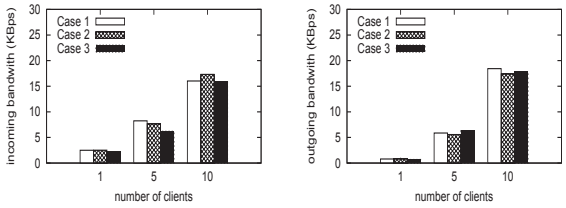
For more realistic evaluation of the zone migration algorithms, we develop a DVE prototype using the Torque 3D game engine[1]. The prototype is a First-Person Shooter multiplayer online game, in which players move around a 3D virtual city, collect resource items and shoot at each other. The game has multiple separate zones, and players will need to select their zones before joining the game. They may also switch from one zone to another during gameplay. Each zone is hosted by a Xen VM running Windows XP Service Pack 3. Multiple zones may run on the same physical server, but each VM only handles a single zone. In this paper, a VM hosting a game zone is referred to as a game server. Figure 4 shows a screenshot of the game.

5.2 Workload model

To conveniently and efficiently evaluate the proposed algorithms with realistic DVE workloads, we implement an artificial, automatic game client to replace real human players. This simulated game client tries to emulate human behaviors during gameplay. For example, it can find its way around the virtual city, collect the resource items, and do some shootings. It sends messages to and receives updates from the game server in the same way as the real game client. We remove the 3D rendering tasks from the simulated game client in order to reduce the load, and thus be able to run multiple simulated clients on one PC.

To ensure that using simulated clients will not affect the outcome of our performance evaluation, we conduct some experiments to verify the load that those clients generate on the game server. We run each individual experiment for 10 times, and report the average result. Figure 5 shows the game server-side incoming and outgoing bandwidth with varying number of clients in three different cases. In the first case, we run each simulated game client on a separate PC. In the second case, all simulated game clients are on a single PC. In the third case, we run real clients with full 3D graphics, each on a separate PC. The results in Figure 5 show that the network workloads in most situations are very similar. In addition, we find that the game server's CPU utilizations (not shown in Figure 5) are also similar in all cases.

The resource consumptions on the client-side PC in the second case (i.e., all simulated clients on a single PC) are not significant either. In particular, CPU utilizations are 1.8% for one client, 4.1% for 5 clients and 5.5% for 10 clients. The data indicate that we may run a number of simulated clients on the same PC without greatly affecting the evaluation.



(a) Incoming (b) Outgoing

Figure 5: Incoming and outgoing game server bandwidth

5.3 Network model

We use netem[2], and the HTB queuing discipline[3] which are included in mainline Linux kernels starting from version 2.4.20 to emulate a WAN with high round-trip latency in our LAN. The current netem version emulates variable latencies, loss, duplication and packet re-ordering, which are typical properties of WANs. On the other hand, HTB allows the use of one physical link to simulate several slower links and to send different kinds of traffic on different simulated links. We use HTB to assign different latency values to client-server and inter-server network links.

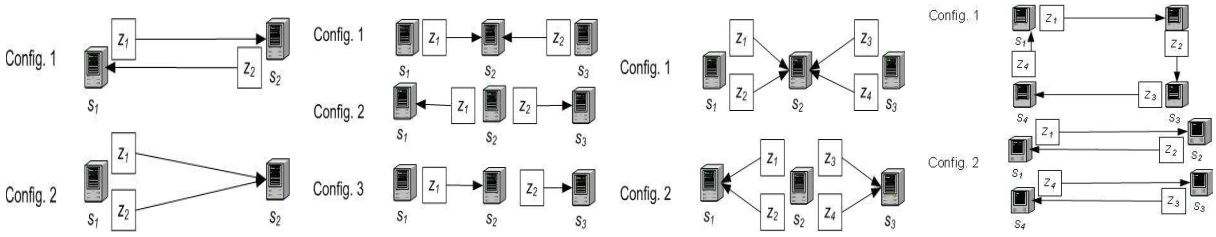
5.4 Default parameters

Unless stated otherwise, the following settings and parameters are used in the experiments. The network bandwidth

[1]http://www.torquepowered.com/products/torque-3d

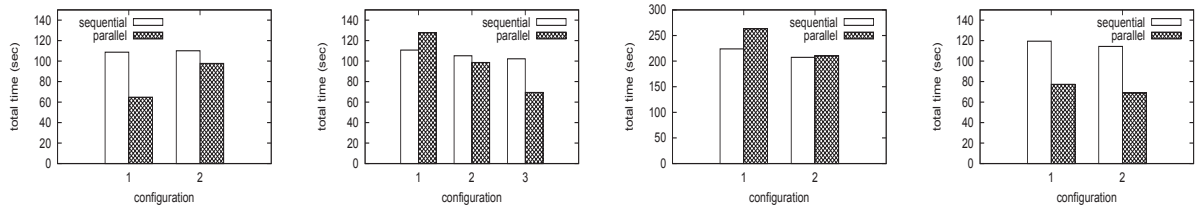
[2]http://www.linuxfoundation.org

[3]http://luxik.cdi.cz/ devik/qos/htb



(a) 2 zones, 2 servers (b) 2 zones, 3 servers (c) 4 zones, 3 servers (d) 4 zones, 4 servers

Figure 6: Zone migration configurations



(a) 2 zones, 2 servers (b) 2 zones, 3 servers (c) 4 zones, 3 servers (d) 4 zones, 4 servers

Figure 7: Completion times for VM migrations

between any pair of physical machines is 100Mbps. The latency between each pair of physical servers hosting VMs is set to 100ms. Client-server latency is equivalent to LAN's latency, which is negligible (around 1ms). There are 10 simulated clients per zone; and all these clients run on the same client-side PC. Each experiment is run for 10 times, and the average result is reported.

6. RESULTS

6.1 Characterization of the remote storage approach

In the first set of experiments, we study the effect of having a remote storage for a VM hosting a DVE zone. Recall that in our virtualization-based zone migration approach, the VM hosting a zone can be freely migrated across physical servers, while its storage may remain at the original server, and only gets migrated if required. Such approach may have some performance issues if the application hosted by the VM is data-intensive, since WANs typically have lower bandwidth and higher latency, compared to LANs. Hence, we need to conduct some experiments to characterize our DVE prototype's storage access patterns; and to carefully study the DVE performance in cases of having a remote or a local storage.

In the experiments, we set up two different scenarios. In the first scenario (referred to as "remote storage") we run a DVE zone on a physical machine, and its storage on another one. The two physical machines are interconnected by a 100Mbps link with 100ms round-trip latency. In the other scenario (referred to as "local storage"), the same DVE zone and its storage are located on the same physical machine.

We first measure the game map loading time when the game server starts up; then the storage access bandwidth for both scenarios during gameplay. For the latter experiment, we use various numbers of clients (1, 5 and 10) playing in the zone for about 2 minutes in each individual measurement. We find that the time to load the game map in case of local storage is around 5 seconds versus 7 seconds for the remote storage. During gameplay, the storage bandwidth consumptions are pretty low, and are similar for both remote and local storage. In particular, storage read operations consume about 1.3 KBps, and do not change much for different numbers of clients. The write operations are negligible in terms of storage access bandwidth. The results indicate that DVE is not a storage-access-intensive application, and having a remote storage would not affect the performance greatly, except when large game maps need to be reloaded frequently.

We also measure the game server latency from the client side in both scenarios. For this purpose, we develop an application-level latency measurement tool similar to qstat[4]. Such tool provides more accurate measurement compared to the simple "ping" command which collects only network-level latency. The obtained results show that there are not much difference in round-trip client-server latencies (around 14-15ms in most cases) in both remote and local storage scenarios, with different numbers of clients.

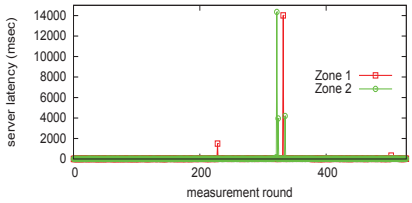
6.2 Zone migration

To realistically examine the performance of the parallel

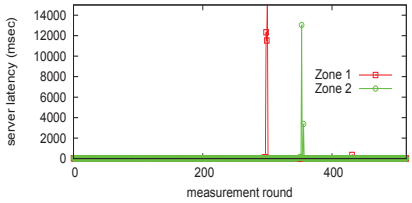
[4]http://sourceforge.net/projects/qstat/

and sequential migration algorithms, we devise a number of different zone migration configurations as shown in Figure 6. The configurations range from 2 zones with 2 physical servers to 4 zones with 4 physical servers. The main performance metric is the migration completion time (in seconds). The experiment results for VM migrations are shown in Figure 7. Generally, the parallel migration finishes faster in most configurations, sometimes significantly faster, e.g., Figure 7(d).

However, surprisingly in some cases such as configuration 1 in Figure 7(b) and Figure 7(c), the parallel migration performs worse than the sequential migration. In these two particular scenarios, we observe that there is only a single physical server as the migrations' destination, and the other two servers are sending all the zones to this destination server concurrently. As a result, the destination server might be overloaded, thus the entire migration process could be slowed down significantly. Therefore, in such cases, the sequential migration is recommended.



(a) Parallel migration

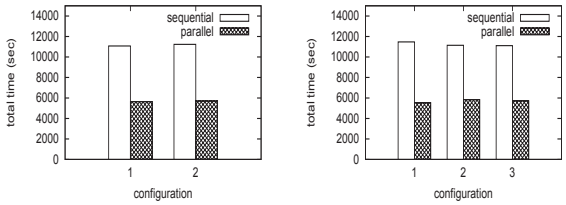


(b) Sequential migration

Figure 8: Client-server latency before, during and after zone migration - 2 zones, 3 servers, config. 1)

Figure 8 shows the client-server latencies for two zones before, during and after zone migration for both parallel and sequential algorithms. We can observe that there is a distinct latency peak for each zone; this happens when the corresponding VM is stopped on the original server and then restarted on the destination server. Despite the high latency during the peak, no game clients get disconnected, and the players just experience a very short "pause" in the game during this transition period.

Figure 9 shows the storage migration completion times for some configurations listed in Figure 6. In our experiments, each VM has a disk image of 10GB. The parallel algorithm outperforms the sequential algorithm in all cases, including cases in which multiple servers send multiple disk images to a single destination server. This is because xNBD applies



(a) 2 zones, 2 servers (b) 2 zones, 3 servers

Figure 9: Completion times for storage migrations

some very small artificial delays into the storage transferring mechanism [8], thus effectively slowing down data transfer speeds. As a result, the destination server is not overloaded with a huge amount of incoming data during parallel storage migrations. As we have mentioned previously, a remote storage does not have much negative performance impact on DVEs, hence there may not be much incentive for migrating the storage as fast as possible.

6.3 Zone mapping

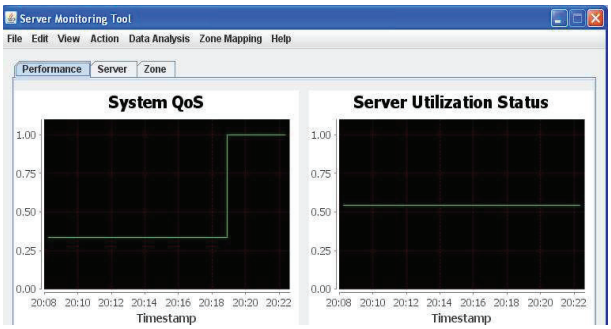


Figure 10: Zone mapping experiment

We use the zone mapping algorithms [16] as a case study to demonstrate the capabilities of the newly proposed zone migration approach. In this experiment, we use 3 zones named z_1, z_2 and z_3 which are initially assigned to 3 physical servers s_1, s_2 and s_3, respectively. We use netem and the HTB queuing discipline to set the client-server network latency between all clients in the system to server s_1, s_2, and s_3 to 50ms, 100ms and 200ms, respectively. We calculate the level of QoS in the system as the percentage of clients that have client-server round-trip latency less than a given threshold, which is 80ms in this experiment.

Figure 10 shows a screenshot captured directly from our DINE tool [15]. We observe that at the start, the system QoS is around 33%, then it jumps to 100%. This is because initially, only the clients in z_1 having client-server latencies that are below the given delay threshold. After zone migrations due to the triggering of zone mapping algorithms, the two zones z_2 and z_3 get migrated to s_1 which has a network latency of 50ms for all clients in the system. Hence, after migration, all the clients in the system have client-server

latencies less than 80ms.

7. CONCLUSIONS

In this paper, we have proposed a virtualization-based approach to address the problem of live zone migration over WAN to support DVE zone mapping or load distribution in a geographically distributed server infrastructure. We have also developed and integrated two algorithms, namely parallel and sequential migrations into an existing DVE performance enhancement and monitoring framework to effectively migrate multiple DVE zones.

Extensive experiments have been carried out with an online multi-player game prototype constructed using the Torque 3D game engine. The results have demonstrated the feasibility of our zone migration approach, and suggested some useful considerations when applying the proposed algorithms in practical settings.

Acknowledgement

This work is supported in part by the Singapore National Research Foundation under Grant NRF2007IDM-IDM002-052.

8. REFERENCES

[1] Distributed replicated block device. *Available at http://www.drbd.org*, Retrieved on Nov 2010.

[2] Kernel-based virtual machine. *Available at http://www.linux-kvm.org*, Retrieved on Nov 2010.

[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the ACM symposium on Operating systems principles*, pages 164–177. ACM, 2003.

[4] P. B. Beskow, G. A. Erikstad, P. Halvorsen, and C. Griwodz. Evaluating ginnungagap: a middleware for migration of partial game-state utilizing core-selection for latency reduction. In *Proceedings of the 8th Annual Workshop on Network and Systems Support for Games*, NetGames '09, pages 10:1–10:6. IEEE Press, 2009.

[5] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg. Live wide-area migration of virtual machines including local persistent state. In *Proceedings of the 3rd international conference on Virtual execution environments*, pages 169–179. ACM, 2007.

[6] C. Clark, K. Fraser, S. Hand, J. G. Hanseny, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *2nd Symposium on Networked Systems Design and Implementation (NSDI 2005)*. USENIX, 2005.

[7] E. Harney, S. Goasguen, J. Martin, M. Murphy, and M. Westall. The efficacy of live virtual machine migrations over the internet. In *Proceedings of the 2nd international workshop on Virtualization technology in distributed computing*, pages 8:1–8:7. ACM, 2007.

[8] T. Hirofuchi, H. Ogawa, H. Nakada, S. Itoh, and S. Sekiguchi. A live storage migration mechanism over wan for relocatable virtual machine services on clouds. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID '09, pages 460–465. IEEE Computer Society, 2009.

[9] A. Iosup, V. Nae, and R. Prodan. The impact of virtualisation on the performance and operational costs of massively multiplayer online games. *IJAMC*, 4(4):364–386, 2010.

[10] K. W. Lee, B. J. Ko, and S. Calo. Adaptive Server Selection for Large Scale Interactive Online Games. *Computer Networks*, 49:84–102, 2005.

[11] J. Lui and M. Chan. An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems. *IEEE Transaction on Parallel and Distributed Systems*, 13(3), 2002.

[12] D. S. Milojicic, F. Douglis, Y. Paindaveine, R. Wheeler, and S. Zhou. Process migration. *ACM Comput. Surv.*, 32(3):241–299, 2000.

[13] V. Nae, A. Iosup, S. Podlipnig, R. Prodan, D.H.J.Epema, and T. Fahringer. Efficient Management of Data Center Resources for Massively Multiplayer Online Games. In *Proc. of ACM/IEEE SuperComputing Conference on High Performance Networking and Computing*, 2008.

[14] S. Singhal and M. Zyda. *Networked virtual environments: design and implementation.* Addison-Wesley, Reading, MA, 1999.

[15] D. Ta, T. Nguyen, S. Zhou, X. Tang, W. Cai, and R. Ayani. A framework for performance evaluation of large-scale interactive distributed virtual environments. In *CIT*, pages 2744–2751, 2010.

[16] D. Ta, S. Zhou, X. Tang, W. Cai, and R. Ayani. Efficient zone mapping algorithms for distributed virtual environments. In *Proc. of ACM/IEEE/SCS PADS*, pages 137–144, 2009.

[17] D. N. B. Ta and S. Zhou. A Dynamic Load Sharing Algorithm for Massively Multi-Player Online Games. In *Proc. of the 11th IEEE International Conference on Networks*, 2003.

[18] D. N. B. Ta and S. Zhou. A Two-phase Approach to Interactivity Enhancement for Large-Scale Distributed Virtual Environments. *Elsevier Computer Networks*, 2007.

[19] F. Travostino, P. Daspit, L. Gommans, C. Jog, C. T. A. M. de Laat, J. Mambretti, I. Monga, B. van Oudenaarde, S. Raghunath, and P. Y. Wang. Seamless live migration of virtual machines over the man/wan. *Future Generation Comp. Syst.*, 22(8):901–907, 2006.

[20] T. Wood, P. Shenoy, K. K. Ramakrishnan, and J. V. D. Merwe. Cloudnet: A platform for optimized wan migration of virtual machines. *University of Massachusetts, Technical Report*, 2010.

[21] S. Zhou, W. Cai, B. S. Lee, and S. J. Turner. Time-space consistency in large-scale distributed virtual environments. *ACM Transactions on Modeling and Computer Simulation*, 14(1):31–47, 2004.