SecondSite: Disaster Protection for the Common Server

Brendan Cully and Andrew Warfield
University of British Columbia and University of Cambridge

Disaster-tolerant systems are complex and expensive constructions that have hitherto been the provision of only the very rich or the very scared. The current state of the art for surviving site-wide failures is to mirror persistent storage to a remote location and cold-start applications there in the event of failure. This requires delicate, tailored modifications to production software to restore application-level consistency before resuming operation.

We propose *SecondSite*, a novel approach to disaster recovery that requires no understanding of, or interaction with, protected applications. Servers are run in virtual machines whose complete runtime state is continuously checkpointed to off-site persistent storage. In the event of a primary site failure, the recovery site (which may concurrently protect multiple locations) immediately activates a consistent snapshot of a group of protected hosts as if they had simultaneously been migrated to the new location. We believe that failure recovery can be made to perform well enough to result in only seconds of downtime — restored hosts should be able to maintain open TCP connections with clients.

As regulatory and environmental changes make disaster recovery necessary in a broadening range of systems, the overhead of application-specific approaches renders them more and more untenable. By running below VMs and capturing whole-system state, *SecondSite* removes the need to modify services to support recovery, and consequently avoids expensive and ongoing development, testing and deployment effort. Moreover, as a platform-level service it can be administered independently of the applications it protects, allowing third parties such as managed hosting providers to exploit economies of scale by sharing the costs of protection among a large number of independent servers, and by compressing redundant data across hosts.

1 Design Goals

To date, virtualization has seen only limited application in disaster recovery. Current commercial offerings[1] use virtualization simply as an administrative container to move machine state to a new location. In these approaches, replication is limited to persistent data, and restoration requires the same reboot and consistency checks that are used in non-virtualized environments. Our approach carries the benefits of virtualization much further by capitalizing on the ability to capture the entire state of a running multi-host system in order to make disaster recovery an automatic property of the virtual environment. The system we propose possesses the following characteristics:

Transparency. Virtualizing the entire hardware infrastructure allows services to be replicated with no change to application, library, or OS code. This makes *SecondSite* applicable to a very broad range of systems, and eliminates the development cost and per-instance complexity of adding disaster recovery code to individual applications.

Speed. Transparent failover can be sufficiently fast to maintain active TCP connections to the outside network. To users of the system, site-wide disaster should be indistinguishable from network delay. By replicating whole-system state and failing over with very low latency, we avoid the need to recover a large window of potentially inconsistent data from persistent storage. This simplifies system-wide recovery and supports our goal of transparency by eliminating the need for application-specific modifications to restore consistency during recovery.

Efficiency. In existing systems, the bandwidth required just to mirror persistent storage across geographically remote locations is considerable, and consistency protocols can incur significant performance overhead; our proposal to replicate live machine state raises these raw costs even further. However, we believe several complementary techniques may be applied in our environment. Redundancy within and across systems may be exploited through content-hashing, caching and compression. Servers that modify state too rapidly to be mirrored in a timely fashion may be throttled, allowing an explicit tradeoff between protection overhead, comprehensiveness, and system performance. Finally, multiple geographically diverse primary sites can share a common backup location, further increasing exploitable data redundancy while amortizing backup facility costs across primary sites.

[1] e.g. http://www.vmware.com/products/vi/vc/ha.html

2 Implementation Challenges

The major challenge faced by a platform-level disaster-recovery system is to maintain an up-to-date and consistent image of each protected group of servers in the face of the limited bandwidth and the unavoidable latency of long-distance network links. In light of the bandwidth constraint, it might seem attractive to run two systems in parallel and tee the external input to both, discarding the output from the backup site. Unfortunately, it is surprisingly hard to guarantee the consistency of such a system: without application-specific knowledge it is difficult to prevent divergence between the two sites due to the accumulation of small differences between the systems (e.g., differences in scheduling or network message arrival times can result in different database transaction orderings). A live backup site also consumes as many resources as the primary even when the primary site is active, and so is far less efficient than a system that instantiates backup servers on demand from stored state.

Our approach is to take frequent distributed checkpoints of protected groups of servers. This is a well-studied problem in its own right, and virtualization facilitates a wide range of synchronization techniques because it can interpose itself on the scheduler and event-delivery systems. To address the potential of visible state difference between the primary and backup sites due to propagation lag, external network traffic can be replicated synchronously through the remote gateway, and the resulting traffic log used to replay the window of outstanding events on the recovering VMs in order to move them forward to the current time.

Capture and Replication Overhead. One drawback of a transparent approach is that it requires significantly more bandwidth, treating protected VMs as black boxes means that all state must be captured. We are exploring several techniques to reduce this to manageable levels, including: sending the deltas of dirty pages; content-hashing pages to discover redundancy; time dilation (slowing down VMs that generate state faster than it can be propagated); overlapping request checkpointing and request handling; ignoring intra-cluster communication (correct snapshots will recreate these messages when executed); and VM introspection or paravirtualization techniques, e.g. marking "hot" pages such as process stacks, whose transmission may be deferred until a snapshot is committed.

In our initial experiments, dirty page compression alone yields an 80% bandwidth reduction for a single VM running the mysql benchmark suite; we expect dramatic improvements in these savings as we improve our implementation. To give an idea of the type of support this makes possible, the aforementioned mysql benchmark currently requires an average of approximately 11 Mbps at a snapshot rate of once per second (this represents something of a worst-case usage model for a managed-hosting provider — web servers are likely to dirty pages at a much lower rate). Furthermore, reduction in snapshot frequency provides better than linear improvement in bandwidth usage: multiple page writes will be coalesced into a single transfer. It is also worth noting that economies of scale should allow managed hosting providers to provision inter-site bandwidth at much lower cost than would be available to end users.

Failover. In order for failover to be successful, network connections must migrate undisrupted across physical locations — a considerably more challenging problem than that of migration in the local area where ARP advertisements may be used to indicate a MAC-level relocation. But if we assume that the managed hosting provider owns both physical facilities and that these facilities are connected to a single, common upstream BGP autonomous system (AS), we may use "dual-homing" for IP fail-over in the same manner that some existing single site configurations do for network link failure protection. The backup site is configured as a dual-homed IP configuration, in which the two "homes" are several hundred miles apart, and we generate BGP updates in response to failure requests, hopefully long before router failure triggers an automatic reconfiguration. We have informally reviewed this proposal both through a discussion on the North American Network Operators Group (NANOG)[2], and through more detailed discussions with commercial hosting providers. The consensus seems to be that similar configurations are used for highly available servers by some customers, and that failover times of a few seconds are certainly reasonable.

3 Conclusion

We believe that it is both possible and desirable to provide transparent disaster recovery services to clusters of unmodified Internet-accessible application servers. Because virtual machines encapsulate server state in its entirety, they make it possible to separate both backup and recovery from the application domain. Thus these services can be provisioned by specialized third parties with the requisite infrastructure, on behalf of a broad range of clients.

Although the scope of this work is ambitious, we anticipate that incremental progress in several areas of the system should deliver interesting and useful results. For example, even when confined to a LAN continuous migration can provide redundancy for entire servers (including transparent recovery when hardware inevitably fails), as well as making possible intriguing forensic tools such as time-traveling distributed debuggers and replayable cluster-wide intrusion logs. Likewise, even infrequent consistent snapshots are useful for recovery of long-running computations of the kind frequently seen in the high-performance computing community. Additionally, wide-area failover is beneficial even without multi-host consistent checkpoints, especially for the many hosted services that exist within a single server.

Those interested can follow our progress at `http://dsg.cs.ubc.ca/secondsite/`.

[2] "*IP failover/migration question.*", June 11, 2006. NANOG archives are available at `www.merit.edu/mail.archives/nanog/`