

MEMORY WRITE TRACKING FOR VIRTUAL MACHINES

BACKGROUND

High availability is a system design approach and associated service implementation that ensures a prearranged level of operational performance will be met during a contractual measurement period. Users want software systems to be ready to serve them at all times. Availability refers to the ability of the user community to access the system, whether to submit new work, update or alter existing work, or collect the results of previous work. If a user cannot access the system, it is said to be unavailable. Generally, the term downtime is used to refer to periods when a system is unavailable.

Virtualization or cloud-computing environments has allowed IT infrastructure to reduce dependence on physical hardware. Yet, virtual machines still run in memory on a physical host. If the physical host reboots or fails, the memory also fails. In a high-availability configuration, this requires that another virtual machine take over providing services. However, any data in the memory of the failing virtual machine will be lost. There is also a time delay while the backup resources are brought on line and start providing service. In the past, work has been done to cut down this failover time for virtual machines by keeping two copies of memory so that a second virtual machine can be started and pickup exactly where the other virtual machine left off. With the growth in memory sizes, keeping two virtual machine's memory in sync across the network has become very difficult.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is an example system diagram showing a primary physical host and a failover physical host.

FIG. 2 is an example system diagram showing a plurality of virtual machine instances that can be used.

FIG. 3 is an example memory synchronization manager having multiple virtual machine memory maps.

FIG. 4 is a flowchart of a method for synchronizing memory between a primary virtual machine and a failover virtual machine.

FIG. 5 is a flowchart of an alternative embodiment for synchronizing memory.

FIG. 6 shows further details of a method that can be performed when synchronizing memory.

FIG. 7 is a flowchart showing a synchronization initialization phase.

FIG. 8 is a flowchart of a method for synchronizing from the perspective of the primary physical host.

FIG. 9 is a flowchart of a method for synchronizing from the perspective of the backup physical host.

FIG. 10 is a flowchart of a method for partial failure of the primary physical host.

FIG. 11 is a flowchart of a method for complete failure of the primary physical host.

DETAILED DESCRIPTION

FIG. 1 is an example system 100 showing a primary physical host 110 and a failover physical host 112. The primary physical host 110 includes a hypervisor 114. As is well understood in the art, the hypervisor presents a virtual operating platform and manages execution of potentially multiple operating systems that can share virtualized hardware resources on the primary physical host 110. An example primary virtual machine 116 is shown executing on the hypervisor 114. When

the virtual machine 116 performs a memory write function, it can communicate through the hypervisor to a physical layer of the primary physical host 100 that includes a hardware-based memory synchronization manager (MSM) 120 and a memory 122. The MSM can include a hash generator 124 (or alternatively a checksum generator) and a memory controller 126. The memory controller 126 can be included within the MSM 120 or positioned separately therefrom. Generally, the memory controller 126 controls the timing needed to complete reads and writes to the physical memory 122. The MSM 120 can communicate with a MSM 130 located on the failover physical host 112 through a network 132. The MSM 130 can include a hash checker 140 (or alternatively a checksum checker) and a memory controller 142, which can be separated from the MSM 130. The memory controller 142 is for writing to a redundant memory 150, which stores a backup copy of memory 122 used by the primary virtual machine 116. The failover physical host 112 also includes a failover virtual machine 160 that can be used should the primary virtual machine 116 fail or otherwise terminates abnormally. As further described below, the failover virtual machine 160 can remain in a paused state until such time as a hypervisor 162 initiates the failover virtual machine 160. Once the failover virtual machine 160 starts, the redundant memory 150 can be synchronized to the memory 122 used by the primary virtual machine so as to make the transition seamless when switching between the virtual machines. In the case of disaster recovery, a persistent storage device 170, such as a solid-state drive (SSD), can be used.

The system 100 of FIG. 1 can use memory change tracking to lessen an amount of data that is synchronized between the primary virtual machine and the failover virtual machine so that the failover virtual machine can immediately take over operations should the primary virtual machine become unavailable. The MSMs 120, 130 can be used to track the changes in the memory 122 and transmit the same to the failover physical host 112 so that the changes can be available, if necessary, by the failover virtual machine 160. The change tracking can be synchronized in nearly real time between the two physical hosts 110, 112, and can be small enough data updates to allow minimal network bandwidth over the network 132. In some embodiments, a change log can be maintained and used to bring the redundant memory 150 up to a synchronized state. In the case of wide-spread failure, memory changes and the base memory footprint can be stored on the persistent storage device 170 to speed off-site recovery.

In some embodiments, the virtual machine's memory can be segmented so that only a critical section of memory is synchronized to further reduce synchronization overhead. The hash checker 140 can be used to rebuild any missing bits on the failover physical host. Additionally, the failover virtual machine 160 can be maintained in a pause state in order to reduce power consumption.

FIG. 2 is a computing system diagram of a network-based service center 200 that illustrates one environment in which a website analyzer can be used. By way of background, the service center 200 is capable of delivery of computing and storage capacity as a service to a community of end recipients. Generally speaking, the service center 200 can provide the following models: infrastructure as a service, platform as a service, and/or software as a service. Other models can be provided. For the infrastructure as a service model, the service center 200 can offer computers as physical or virtual machines and other resources. The virtual machines can be run as guests by a hypervisor, as described further below. The platform as a service model delivers a computing platform