High Performance Virtual Machine Recovery in the Cloud

Valentina Salapura[1] and Richard Harper[2]
[1]IBM T. J. Watson Research Center, 1101 Kitchawan Rd, NY, Yorktown Heights, U.S.A.
[2]IBM T. J. Watson Research Center, Research Triangle Park, NC, U.S.A.

Keywords: Cloud Computing, High Availability, Virtualization, Automation, Enterprise Class.

Abstract: In this paper, we outline and illustrate concepts that are essential to achieve fast, highly scalable virtual machine planning and failover at the Virtual Machine (VM) level in a data center containing a large number of servers, VMs, and disks. To illustrate the concepts a solution is implemented and analyzed for IBM's Cloud Managed Services enterprise cloud. The solution enables at-failover-time planning, and keeps the recovery time within tight service level agreement (SLA) allowed time budgets via parallelization of recovery activities. The initial serial failover time was reduced for an order of magnitude due to parallel VM restart, and to parallel VM restart combined with parallel storage device remapping.

1 INTRODUCTION

Cloud computing is being rapidly adopted across the IT industry as a platform for increasingly more demanding workloads, both traditional and a new generation of mobile, social and analytics applications. In the cloud, customers are being led to expect levels of availability that until recently were available only to the largest of enterprises.

Cloud computing is changing the way high availability (HA) of a data center can be implemented. It is widely recognized that the standardization, virtualization, modularity and cross system management capabilities of cloud computing offer a unique opportunity to provide highly resilient and highly available systems. Resilience techniques can build on a well-defined and uniform framework for providing recovery measures for replicating unresponsive services, and recovering failed services to respond to disaster scenarios. Since virtualization allows packaging of workloads — operating system, applications, and data — into a portable virtual machine image container, it facilitates transfer of workloads from one server to another. High availability features can migrate a VM image from one physical server to another within the same data center if the original server suffers any failure, performance loss, or to perform scheduled maintenance.

However, clouds and the workloads that run on them are big. Many high availability systems were originally designed for smaller managed environments, and do not scale well as the system size and complexity increases. Detecting failures, determining appropriate failover targets, re-mapping storage to those failover targets, and restarting the virtual workload have to be carefully designed and parallelized in order to meet the service level agreement (SLA) for large systems.

This paper describes a highly scalable parallel virtual machine planning and recovery method that enables high availability at the Virtual Machine (VM) level for large data centers comprising many high-capacity servers, many VMs, and a large number of disks in a storage area network (SAN). The system enables on-the-fly failover planning and execution for a compute environment with a large number of servers and storage devices.

The functionality described in this paper has been released as part of IBM's enterprise cloud offering known as CMS (Cloud Managed Services), where it was used to provide scalable HA for the AIX Logical Partitions (LPARs) running on the CMS Power Systems (Sinharoy *et al.*, 2015) servers. To stay within this context, the paper will continue to use the Power LPAR terminology. However, the concepts described here apply equally well to any platform that is similarly structured. While in this paper we focus only on the infrastructure level resiliency, CMS cloud implements all application level high availability approaches. However, they are not in scope of this paper, and will not be discussed here.

Salapura V. and Harper R..
High Performance Virtual Machine Recovery in the Cloud.
DOI: 10.5220/0005493405590564
In *Proceedings of the 5th International Conference on Cloud Computing and Services Science* (CLOSER-2015), pages 559-564
ISBN: 978-989-758-104-5

2 BACKGROUND AND POSITION STATEMENTS

2.1 Virtual Machine-Level and Application-Level High Availability Are Complimentary

There are multiple approaches to provide a high availability solution in a virtual environment. One approach is to provide HA at the application level, using what are commonly known as HA Clustering techniques. Another approach is to provide availability at the infrastructure level, using VM-level HA.

Application-level high availability techniques are built around application clustering technology. These solutions are used to improve the availability of applications by continuously monitoring the application's resources and their physical server environment, and invoking recovery procedures when failures occur. These solutions typically use multiple virtual machines which are working together in order to ensure that an application is always available. These VMs are arranged in active-passive or active-active configuration. When one VM fails, its functionality is taken over by the backup VM in the cluster. Examples of these solutions are IBM PowerHA (IBM, 2008), Microsoft Clustering Services (Microsoft, 2003), Veritas Storage Foundation, and LinuxHA.

HA solutions at the infrastructure level are designed to ensure that the virtual resources meet their availability targets. This is accomplished by continuously monitoring the infrastructure environment, detecting a failure, and invoking recovery procedures when a failure occurs. Typically, such recovery procedures involve restarting the failed VM, either on the same or a different physical server.

Although this paper will not discuss application-level HA in detail, we have found that application-level HA and infrastructure-level HA can operate beneficially together with no mutually destructive effects. A tidy separation of concerns exists - infrastructure-level HA restarts VMs when appropriate (sometimes on alternate servers), while application-level HA sees these restarts as simple system crashes and recoveries, which it is designed to tolerate anyhow. In addition, recovery of the VMs in a cluster on another server after the originating server fails restores the redundancy that the application-level HA cluster relies upon, minimizing the time during which that cluster is operating with degraded resiliency.

2.2 Dynamic Storage Mapping Is Preferable to Static Mapping

Virtualized infrastructures can be designed such that either all physical servers in a server pool are statically mapped to all the storage devices that may be used by the virtual machines, or all physical machines are dynamically mapped to only the storage devices that are needed to support the virtual workload running on the respective physical machines. The first design choice has the merits of being simpler to operate, since no remapping of storage is required as virtual machines migrate or failover within the pool. However, it is unsuitable for high-scale cloud environments where the pool may consist of hundreds or more servers, supporting thousands of virtual machines, which in turn use even more storage devices. In this environment, the architectural and design limits of the hypervisor running on each physical server cannot support the huge number of simultaneous connections required to support all possible VM-storage device mapping. Instead, it is desirable to have a physical server only possess storage mappings for those VMs that are actually running on that physical server, and this is the design point utilized in this paper. The disadvantage of this approach are that, if it is necessary to migrate or failover a VM from one server to another, it is necessary to map that VM's storage to the destination physical server, and unmap that storage from the source physical server.

2.3 Parallelization of Recovery Is Critical to Maintaining SLAs

Complex recovery activities consist of a number of sequential steps that must often be executed using tools, processes, and infrastructure elements that have limited recovery performance and concurrency. Given the large scale of a recovery operation (recovery of potentially thousands of virtual machines across dozens of physical servers), it is absolutely necessary to judiciously parallelize these recovery actions and eliminate bottlenecks to meet tight SLAs. The limited space herein does not permit a full exposition of these position statements, but we will partially illustrate them using an implemented case study based on the IBM Cloud Managed Services (CMS) architecture.

3 CMS POD ARCHITECTURE

CMS is a cloud computing offering for enterprise customers. It is designed to bring the advantages of cloud computing to the strategic outsourcing customers of IBM. It provides standardized, resilient, and secure IBM infrastructure, tools, and services with full ITIL management capabilities (Cannon, 2011). CMS offers functions such as consumption-based metrics and automated service-management integration.

The design of the CMS is based upon a unit called the point of delivery (PoD). A PoD contains many physical managed resources (server, storage, and network) that are virtualized, and provided to customers as an infrastructure offering. A CMS PoD contains Intel-based servers to support virtual and bare metal Windows and Linux workloads, and IBM Power servers to support virtual AIX workloads. The Power virtual machines are called Logical Partitions, or LPARs. This paper focuses on the recovery of the AIX workloads, contained in LPARs, in the event that a Power server fails.

A PoD is designed to be highly available, with the physical infrastructure architected to eliminate single points of failure. The customer is offered selectable availability SLAs, which are contractual obligations and may include penalties for noncompliance. These availability agreements are only for unplanned outages and refer to Virtual Machine availability. CMS supports multiple levels of availability ranging from 98.5% to 99.9%. A more detailed description of the CMS can be found in (Salapura, 2013).

PoDs also contain a number of managing servers which host management tools for storage management, backup, and performance monitoring.

3.1 Fault Model: Permanent Failure of a Power Server

The remainder of this paper will describe the architecture we have created for recovering LPARs on other physical servers when one or more Power Systems physical servers hosting those LPARs has failed.

In this failure mode, a Power Server suffers a hardware failure from which it cannot recover in a short time (for example, 10 minutes) and for which maintenance/repair is required. In this case, the failover process will restart all affected LPARs on another Server. The function implementing this recovery process is called Remote Restart. The recovered LPARs need to use the same network storage disks – referred to as LUNs (logical unit number) that the original Server was using. Restarts are prioritized by SLA. Recovery from other types of outages and transient failures are covered by means not described in this paper.

4 REMOTE RESTART ARCHITECTURE

The architecture of the Remote Restart solution used in CMS PoDs is illustrated in Figure 1. There are one or several managing servers, indicated in the upper part of the figure, and a number of managed servers with storage are illustrated in the lower part of the figure. The managing servers host tools for controlling, provisioning, managing and monitoring of the workload on managed servers. Relevant managing tools are Provisioning engine, which uses a DB to maintain all the PoD management information, a Storage management engine, and a Hardware Maintenance Console (HMC) for server management. The Remote Restart software and collected configuration data resides on a management server for Virtualization management.

The managed servers host LPARs running customers' AIX workload. Each managed Power server also contains dedicated LPARs called Virtual I/O Servers (VIOS) that virtualize external storage and present it to the customer's LPARs.

4.1 Overview of Recovery Procedure

The tasks that the Remote Restart solution performs are as follows:

Periodic data gathering and persistence: configuration and status of LPARs in a PoD is collected periodically. The time interval for data gathering is configurable, and is given later in this paper. There are two sources of collecting needed information:

- information about physical servers in the PoD, all LPARs and their hosts, and their storage and network configuration; this information is collected via HMC;
- SLA availability information for all LPARs; this information is obtained by querying the Provisioning engine database.

Server failure detection: the health of all servers in a PoD is monitored in order to detect their failure. A failure of a server is detected via HMC when it returns an ERROR state.

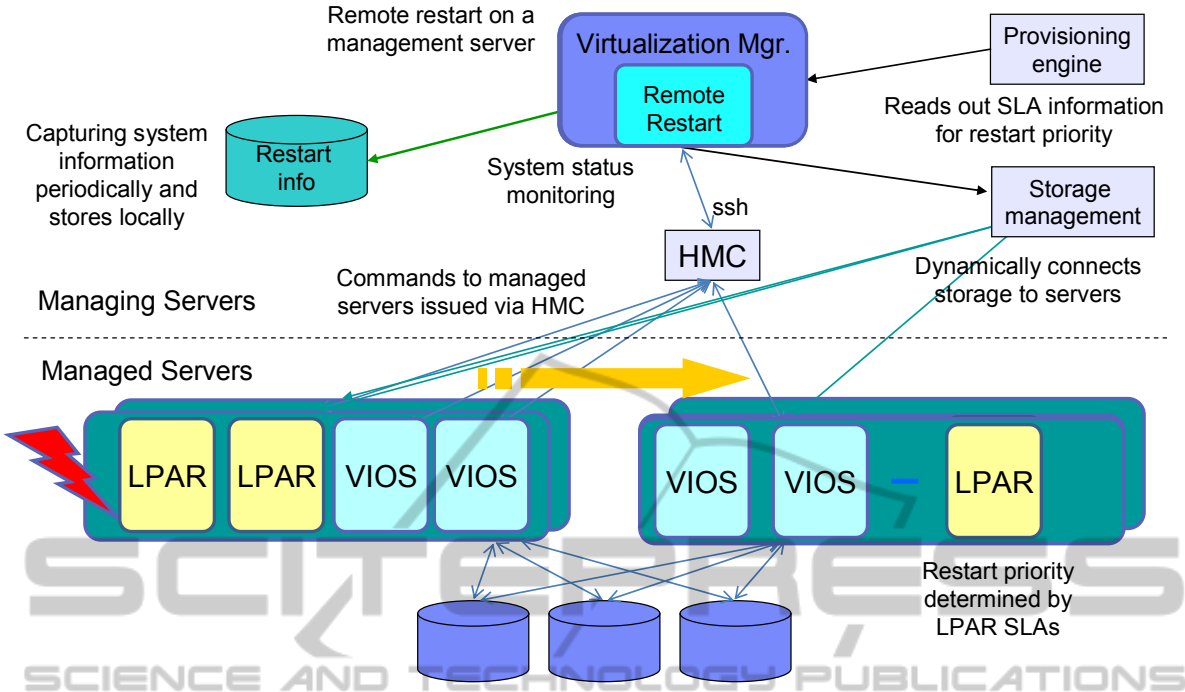

Figure 1: Remote Restart architecture.

Server fencing: once a server is determined faulty, it is powered off via HMC commands.

Failover planning: provides an evacuation plan. Our Remote Restart implementation uses a "Dynaplan" (Harper, 2011) algorithm to determine the optimal failover targets.

VIOS configuration for failover: in this step, virtual SCSI devices are created via HMC on the failover server for LPARs to be restarted.

SAN configuration for failover: LUNs are not connected to all servers in a PoD, and the connecting of LUNs to the failover servers according to the evacuation plan is performed in this step.

LPAR restart: once virtual SCSI devices are created and LUNs and connected to the failover server, an LPAR is restarted on the failover server via HMC commands.

The Remote Restart scripts performs these steps by issuing `ssh` commands to the HMC, via database queries to the Provisioning engine, and by issuing commands for storage configuration.

4.2 Failover Planner

Failover planning is based on a parallelized algorithm evolved from the prior dynamic resource planner described in (Harper, 2011). The planner formulates a schedule to restart a large collection of interdependent VMs on a large collection of resources. There are a number of constraints the planner has to meet, for example that recovery time objective is met, that the maximum number of the most important dependency groups is started, that VMs within a dependency group are started in the proper order, and that the capabilities of the environment (e.g., restart bandwidth and capacities) are not exceeded.

Restart priority is a partial ordering of all VMs into priority classes. Within a given priority class, all VMs can be restarted in parallel, subject to restart parallelism constraints of the physical environment and application start order dependencies. A "restart rules" language allows customization of the restart priority based on restart rules. A restart rule template can be automatically populated by discovery tools and/or manually edited.

The restart priority is automatically and dynamically determined based on a number of VM properties, such as SLAs, application priority, application topology, and other rules as determined by the dynamic restart priority calculator and a given set of rules. Priority aggregation rules convert the various restart rules into the VM restart partial priority order while taking into account application dependencies.

The cost of running the planner is low, so it is run at failure-handling time. In addition, the failover

planner is run once per day for each server in a PoD to determine any resource constraint, for example to determine if there are capacity problems so that not all LPARs can be hosted on the remaining hosts. If this condition is detected, a warning notification is sent to the cloud administrators for the purposes of planning.

5 IMPLEMENTATIONS AND RESULTS

5.1 Initial Implementation: Serial Restart

The restart priority of LPARs is based on their SLA. Thus, in case of failover, the highest SLA workloads would be restarted first followed by the next highest SLA. Within the same SLA level, restart priority is random. In an early CMS release, restart capability was needed only for workloads with the two highest level SLAs. This initial Remote Restart implementation was implemented as a single process which, after the failure of a server is detected, and the need for a failover process was determined, would initiate the failover process.

For each LPAR on the affected server, the failover planner determines a destination server, and the restart process starts. The failover process is performed for the highest priority LPARs first, configuring the storage and network for these LPARs to their destination servers, and restarting them at the destination server. After all LPARs with the highest restart priority are restarted at their target servers, the next lower priority level LPARs are processed.

There are two significant time components to executing the restart. The first is the process of unmapping the LUNs from the (failed) original server and mapping them to the designated failover server. This time is proportional to the number of LUNs connected to the LPAR. The second time component is the process of restarting the LPAR on the designated failover server.

In this early CMS release, each LPAR was allowed to have up to two LUNs. For the case where only the top two SLAs were to be restarted, with up to two LUNs per LPAR, the SLA time budget was readily met.

However, in the subsequent releases of CMS, the number of disks per LPAR was continuously increased. In addition, it was necessary to extend restart capabilities to all SLA levels. With these increases, it was clear that we needed a solution for Remote Restart which would handle restarts for a larger number of LPARs containing more LUNs, within the SLA time limits.

5.2 Parallel Restart

The requirement for an increased number of LUNs per LPAR, and the increased number of LPARs which need to be restarted motivated us to improve the Remote Restart solution using parallel processes. We chose to use server-level parallelism in which the level of parallelism depends on the number of operational servers in the PoD.

In our parallelization scheme, one restart process is launched for each destination server. For example, in a PoD with 6 servers, and one failed server, there would be up to 5 destination failover servers. One restart process is initiated for each destination server. LPARs assigned for restart on that particular server are restarted sequentially, starting with the highest priority LPARs in that group. For each LPAR, storage is mapped, storage and network drivers are reconfigured for the target server, and the LPAR is restarted at the destination server. Once all highest priority LPARs assigned to that destination server are restarted, the next SLA priority level LPARs are processed. A similar process is performed in parallel for all destination servers.

These parallelization steps ensured that the failover time was well within the allowed SLA for the subsequent releases of CMS.

5.3 Parallel Disk Mapping

However, the disk capacity in CMS continues to increase. For the current release, each LPAR can have up to 24 LUNs and up to 96 TB of storage. For a large number of LPARs on a single server, this can lead to the case where a very large number of storage LUNs has to be mapped to different servers in short time.

Analysis indicated that the procedure that was taking the most amount of time was the process of mapping disks to the destination server, so our next improvement focused on parallel disk mapping. In this implementation, in addition to the number of parallel failover processes that is started, we also initiate the mapping of multiple disks attached to a single LPAR in parallel. We limit the number of simultaneous mappings of disks for a single failover stream to four to avoid potential bottleneck at the storage management interface. By measuring the time needed for restarting individual LPARs with a

different number of LUNs and time measured for parallel failover streams, we analyzed failover time needed for parallel disk restart.

The analysis shows that adding this additional level of parallel processing brings the failover time requirements well within the available time budget for the worst-case configuration known to date.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we presented a highly scalable parallel virtual machine planning and failover method that enables high availability at a VM level in a data center. This solution is efficient for large data centers comprising many high-capacity servers, many VMs, and a large number of disks. The solution is implemented and used in IBM's CMS enterprise private cloud.

The system enables at-failover-time failover planning and execution for a compute environment with a large number of servers and storage. The described system keeps the recovery time within limits to a service level agreement (SLA) allowed time budget. With this design, we reduce the initial failover time requirements by more than an order of magnitude by using parallel failover and parallel storage mapping implementation.

As our future work, we plan to explore the applicability of this solution for disaster recovery (DR), where a whole PoD needs to be restarted at a failover data center within the allowed recovery time objective (RTO).

REFERENCES

Cannon, D. 2011. ITIL Service Strategy 2011 Edition, The Stationery Office, 2nd edition, 2011.

Harper, R., Ryu, K., Frank, D., Spainhower, L., Shankar, R., Weaver, T., 2011. DynaPlan: Resource placement for application-level clustering, 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops, pp. 271 – 277, 2011.

IBM, 2008. Implementing PowerHA for IBM i, IBM Corporation, Armonk, NY, USA, 2008. [Online]. http://www.redbooks.ibm.com/abstracts/sg247405.html.

Microsoft, 2003. Introducing Microsoft Cluster Service (MSCS) in the Windows Server 2003 Family, Microsoft Developer Network. [Online]. https:// msdn.microsoft.com/en-us/library/ms952401.aspx.

Salapura, V., Harper, R., Viswanathan, M., 2013. Resilient cloud computing, IBM Journal of Research and Development, vol. 57, no. 5, pp. 10:1-10:12, 2013.

Sinharoy, B. *et al.,* 2015. IBM POWER8 processor core microarchitecture. IBM Journal of Research and Development, vol. 59, no. 1, pp. 2:1-2:21, 2015.