2/3/2016

High Availability with KVM for IBM z Systems

Bob Abrams
abrams@us.ibm.com
Scott Loveland
d10swl1@us.ibm.com
Reinhard Buendgen
Buendgen@de.ibm.com
Viktor Mihajlovski
mihajlov@de.ibm.com
Steven Cook
cooksd@us.ibm.com

IBM Corporation

Special Notices

Please Note: Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary.

Trademarks

Table of Contents

Abstract 4
Introduction 4
Why host your Linux guests on IBM z Systems? 6
Summary of KVM for IBM z High Availability Recommendations 8
High Availability Best Practices 8
Building a KVM High Availability Environment 10
Where is a good place to start? 11
KVM High Availability: Avoiding Single Points of Failure 12
Linux Guest High Availability 15
KVM Configuration Scenarios 15
Disaster Recovery (DR) 17
Summary 18
References 18
Acknowledgements 19

Abstract

KVM for IBM z Systems hosts Linux on z Systems guest virtual machines while benefiting from the high reliability, availability and serviceability characteristics of the z Systems hardware. The KVM for IBM z offering is optimized for z Systems architecture and provides standard Linux and KVM interfaces for operational control of the environment, as well as supporting OpenStack interfaces for virtualization management. This white paper focuses on considerations for setting up a highly-available KVM for IBM z environment hosting your business applications. With IBM's investments in z Systems reliability, availability and serviceability, z Systems is an excellent platform to host a High Availability KVM for IBM z environment.

Introduction

KVM for IBM z Systems provides open source virtualization for IBM z Systems and the LinuxONE platforms. Using the combination of KVM virtualization and IBM z Systems and LinuxONE, you can host differing Linux workloads while deploying fewer systems and enjoy the performance and RAS (Reliability, Availability, Serviceability) characteristics of the z Systems environment.

KVM for IBM z Systems leverages Linux on z Systems components to support hundreds of guest operating systems in each z Systems partition[1]. The components include the KVM kernel module, a guest implementation (via QEMU), libvirt remote virtualization management and a variety of error handling. At the same time, availability functions developed specifically for Linux on z are utilized, such as dynamic I/O reconfiguration.

The goal of configuring for high availability (HA) is to provide continuous service for your business applications, while masking unplanned outages from end-users. High availability requires that single points of failure be identified and eliminated by configuring redundant instances, possibly with load balancing components to redirect workload to alternate paths based on availability and capacity, while protecting data consistency. High availability employs technologies that automate failure detection and recovery processing automatically to minimize outage windows, with little or no impact to guest operating systems or business applications.

However, even in the best HA solution, a recovery event can be a distressing occurrence. It can cost time, resources and could affect the reliability of the solution (especially if the failure occurs in the second redundant component). Therefore it is best to host a High Availability environment on systems that are reliable to begin with. The z Systems

[1] A partition is a subset of a computer's hardware resources, virtualized as a separate computer. In effect, a physical machine can be partitioned into multiple logical partitions, each hosting a separate operating system. On z Systems, partitions (often called Logical Partitions, or LPARs) are managed by a firmware component called PR/SM. Each KVM for IBM z Systems instance is hosted in a z partition, and each KVM hosts Linux guests in its partition.

platform is the perfect Linux/KVM host given IBM's investments in RAS for z Systems, as described in the next section.

With an awareness of the Linux components that support KVM, the system administrator can configure an environment that takes advantage of the high levels of hardware, firmware and functional redundancy, even when configured to run across multiple partitions (each hosting a KVM hypervisor) in a single physical server! Middleware such as Websphere Application Server Network Deployment and DB2 HADR have built-in clustering technology that enables cluster members to span guests and operate in either an active-active or active-standby fashion. Furthermore, OpenStack can be configured to spread guest images across z Systems partitions, viewing each partition as a separate hypervisor.

If you are reading this white paper, you have likely chosen KVM for IBM z to host Linux guests on IBM z Systems, and wish to find out more about how KVM and Linux on z Systems can host High Availability environments supporting your business applications. In particular, your business likely fits one of the following scenarios:

- **New to z Systems with an existing investment in KVM and Linux guests on distributed systems**: Your installation wishes to host key Linux business applications on a mainframe system, inheriting z Systems platform qualities service, by moving existing KVM-managed Linux guests from a distributed environment to KVM and Linux on z Systems. Furthermore, your enterprise can now easily integrate Linux servers into your existing server farm infrastructure and cloud offerings, while gaining the benefits of virtualization and scalability on z Systems. Your goal may be to reduce and eventually eliminate the costs of a distributed server farm.
- **Setting up a cloud environment:** Your installation is setting up a cloud environment to host large scale Linux servers supporting business-critical applications.
- **Integrating with a heterogeneous application:** Your installation currently has a heterogeneous business application, where distributed KVM is the 'primary", and wishes to set up System z as a consumer.
- **Integrating KVM for IBM z into existing z environments:** Your installation hosts only z/OS environments in a physical server, and wishes to introduce virtualized Linux guests running in one or more z partitions in support of other business applications, with the two workloads running (and perhaps interoperating) side by side in the same physical server.

Each of these scenarios requires knowledge about KVM and its functions, as well as the Linux ecosystem that supports it. This paper starts with a description of the differentiating features that z Systems hardware and firmware bring to KVM for IBM z and Linux on z Systems. The KVM ecosystem is described, including the stack of components that impact high availability characteristics. KVM configuration scenarios are then described, with a discussion on the advantages and limitations of each. Disaster Recovery is included in this topic as well.

Why host your Linux guests on IBM z Systems?

Linux and KVM for IBM z Systems inherit all of the benefits of the System z hardware. The historic strengths of the z mainframe (reliability, availability and serviceability) when aligned with the stability and openness of Linux, provides a strong platform for hosting your business applications on Linux servers.

Consider the following:

- **Hardware-enabled virtualization** allows KVM for IBM z to host many guest virtual servers in a single logical partition (LPAR), while supporting multiple LPARs on a single mainframe system, by sharing resources such as CPU and device adapters across virtual servers. This provides an environment where multiple Linux servers hosted by KVM for IBM z can do the job of many distributed systems in an IT enterprise. In addition, memory and CPU over-commitment further enables hosting of guests at a very large scale. The high scalability also reduces the number of physical servers and amount of networking infrastructure, reducing the overall Total Cost of Ownership (TCO) … not to mention savings in cooling, maintenance, power, software support and infrastructure costs. The z Systems hardware is designed from the ground up to optimize virtualization.
- **Security:** Linux security and integrity features are further extended on z Systems by integrating hardware-based security features into the operating environment. This includes encryption and cryptographic solutions to help secure data from leakage threats. The Common Criteria certification of the z Systems HW partitioning (LPAR) according to EAL 5 ensures that two KVM for IBM z hosts running in different LPARs on the same physical system are fully isolated.
- **High Availability:** Multiple partitions deliver very high levels of hardware and firmware redundancy, allowing each to be an excellent host for your Linux guests. Similarly, high availability of KVM for IBM z Systems guests can be achieved by leveraging Linux resource management (bonded network adapters, multi-path disk access, etc.). Middleware such as Websphere Application Server Network Deployment or DB2 HADR support automated failover to alternate KVM guests in the defined cluster.
- **Input/Output (I/O) virtualization:** KVM for IBM z Systems I/O virtualization enables sharing of physical I/O resources among virtual servers, supporting an array of virtualized I/O configurations, including Fiber Channel (FC) and

Enhanced Count Key Data (ECKD[2]) storage and OSA[3] network cards. This includes:
  - Sharing of physical I/O resources
  - Dynamic addition & deletion of virtual I/O devices, eliminating downtime to modify I/O device configurations for virtual servers
- **Live virtual server migration:** KVM for IBM z Systems supports live migration of virtual servers between different KVM for IBM z hosts running in different partitions that may or may not be located on different IBM z Systems servers allowing business applications to remain active during a scheduled outage of a hypervisor image.
- **Management capabilities** of KVM for IBM z Systems are provided by libvirt APIs, which enable command line interfaces (CLIs) to be used to administer the hypervisor resources and virtual machines. Furthermore, KVM for IBM z can be monitored and administered using open source tools such as Nagios, Ganglia, and OpenStack.
- **Hardware-enabled availability features:** Linux for z Systems also supports high availability features provided by the z Systems and LinuxOne hardware such as dual power supplies, dynamic firmware updates, dynamic I/O reconfiguration, First Failure Data Capture (FFDC) collection/analysis, error isolation, recovering from machine checks and I/O errors and the ability to dynamically vary devices on and off.
- **Memory robustness:** Computers have historically had memory DIMM problems. The newer "RAIM memory" technology eliminated server failures due to memory crashes. In the last three generations, since IBM introduced RAIM memory technology, z Systems have had no server failures due to memory crashes. Also, z Systems have had memory DRAM failures – IBM sells memory by the terabyte. But the operating systems aren't sensitive to that because IBM included up to 1200 spare DRAM chips in a server (depending on how many DIMM are installed). DRAM continues to fail, but we rarely replace memory DIMM for DRAM failures, maybe once in a generation.
- **Concurrent repair:** Linux running in a z Systems server inherits the full z Systems RAS, the hardware and firmware recovery, error checking, diagnostics, tracing and concurrent repair. Hardware failures do not mean that the operating system, or the applications, noticed. 97%[4] of the repair actions are fully concurrent. That means mobile applications can keep accessing their data, databases keep warehousing data, and sales terminals keep selling.

---

[2] The ECKD storage architecture in Linux defines disks formatted as "arrays" of fixed size (4K byte) blocks. ECKD refers to a set of channel commands (collectively Channel Command Words, CCWs) that are accepted by a disk subsystem employing the ECKD recording format.
[3] The Open Systems Adapter (OSA) is a network controller installed in an IBM z Systems mainframe I/O cage. The adapter integrates several hardware features and supports many networking transport protocols. For more information, see the OSA-Express Implementation Guide, www.redbooks.ibm.com/abstracts/sg245948.html
[4] Based on IBM field data

Summary of KVM for IBM z High Availability Recommendations

Before getting into the details of the KVM for IBM z environment, and its high availability considerations, here is a summary of High Availability recommendations as a consolidated list.

Hosting your KVM for IBM z hypervisors:

- Avoid single points of failure. Replicate control units, I/O paths, I/O & network ports and adapters (FICON, FCP, OSA), FICON directors, fiber channel switches, network routers, SAN fabric, data and z13 partitions hosting your KVMs.
- Deploy a journaling file system, such as ext4
- Deploy hardware data mirroring using the SAN Volume Controller (SVC) or IBM System Storage DS8000 disk with consistency groups. SVC handles mirroring at the disk controller. Consistency groups control the replication of group of physical disks, maintaining data consistency across all disks.

Hosting your KVM for IBM z guest virtual machines:

- Where applicable, deploy a load-balancing solution, such IBM's Websphere Edge Components for KVM for IBM z guests.
- Deploy application and middleware clustering, such as Websphere Application Server Network Deployment (WAS-ND) or DB2 HADR
- Define a copy of the KVM for IBM z guest configuration on secondary storage for failure recovery by a second partition.
- Configure autostart for guest virtual machines using *virsh* commands
- Deploy virtual IP addressing and dynamic routing for guest routing
- Active/standby or Active/active configuration for KVM for z Systems guests

High Availability Best Practices

Before discussing KVM High Availability, it makes sense to review the Availability definitions used by IBM (1):

- **High Availability (HA)** – Provide service during defined periods, at acceptable or agreed upon levels, and mask *unplanned* outages from end-users. It employs fault tolerance, automated failure detection, recovery, bypass reconfiguration, testing, and problem and change Management.
- **Continuous Operations (CO)** -- Continuously operate and mask *planned* outages from end-users. CO employs non-disruptive hardware and software changes, non-disruptive configuration, and software coexistence.
- **Continuous Availability (CA)** -- Deliver non-disruptive service to the end user seven days a week, 24 hours a day (there are no planned or unplanned outages).

Compare this to **Disaster Recovery**, which focuses on the delivery of non-disruptive service after a disruptive event, usually over long distances. This typically employs a multi-host or multi-site configuration with sufficient network configuration, data mirroring and error detection to resume business with little delay.

Now let's consider some High Availability best practices:

- The number one best practice when constructing a High Availability environment is to avoid single points of failure in order to minimize the impact of any individual failure. Replicate everything! z13 partitions hosting your KVMs, control units, I/O paths, ports and cards (such as FICON and FCP host adapters and OSA network adapters), FICON directors, fiber channel switches, network routers, disks and data. There's a lot to consider, yet this is similar to the structure that many have built for Linux environments.
- Of equal importance is the need to protect data consistency to ensure minimal, or no data loss, and able to return your file systems and data bases to a consistency point after a system or component crash. This is done using journaling to quickly recover databases and file systems to the point of failure; and file system and database mirroring to allow processing to continue on a surviving system. The method of mirroring also deserves careful consideration, with approaches such the SAN Volume Controller (SVC) or IBM System Storage DS8000 that handle mirroring at the disk controller (with little or no granularity), and software mirroring approaches that provide control over what gets mirrored.
- Since you'll have replicated execution environments, you need to put a good load balancer in place for stateless workloads to redirect requests based on available capacity.
- Storage redundancy is perhaps the most complex area to evaluate. Simple variations include
  - Shared storage, where replicated KVM hosts are booted from the same disk & data as prior to the outage
  - FICON attached disks, such as ECKD and multi-channel connections
  - FCP / SCSI attached disks with multi-pathing.
  - Virtual disk mirror (SAN volume controller); LVM mirror
- The last of the important HA considerations is the need to automate failure detection and failover, facilitating recovery actions intended to minimize outage windows. Most automation functions are based on cluster monitoring and management services that detect failures and initiate failovers.

Compare these considerations with designs for Continuous Operations, which focus on making host changes without impacting any workload running on the system. The most popular approach is to use KVM's live guest relocation to transfer guests to a different hypervisor, install and configure changes to the original hosting environment and migrate the guests back to the original KVM host. In addition, there is a library of "*virsh*" commands that are available to query and dynamically modify KVM guests.

Building a KVM High Availability Environment

The KVM hypervisor allows multiple guest operating systems to share a single hardware host. The hypervisor creates virtual machines (Linux guests) by coordinating requirements for processor, memory, hard disk, network, and other resources through the host operating system.

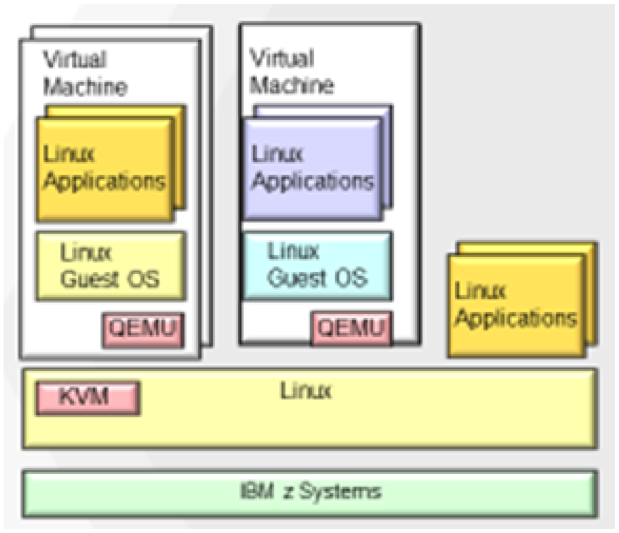

Figure 1: Linux/KVM environment

The Linux operating system with KVM (kernel module) support runs in a z Systems partition. The IBM z13 supports up to 85 partitions, each configured to host an operating system with specified capacity, I/O and network capabilities. The KVM kernel module provides the hardware virtualization for each Linux guest, including virtual CPU and memory definitions. In addition, the KVM kernel provides file system and shared disk access. The QEMU component virtualizes the I/O resources for each guest, sharing the underlying hardware (processors, memory) available in the hardware partition, with each deployed Linux guest. QEMU also supports Live virtual server migration, which enables workload migration between KVM for IBM z hosts with minimal (or no) impact. Each virtual machine is a QEMU process, configured with virtual CPUs, memory, storage and network resources and hosts a Linux guest operating system, along with any Linux applications running in that guest operating system. The structure is basically the same as for distributed Linux/KVM systems, while inheriting z Systems first class RAS characteristics.

When virtualizing Linux guests, KVM for IBM z leverages the Linux ecosystem to support hundreds of guest operating systems in each z Systems partition, while inheriting the RAS capabilities described earlier. The ecosystem includes the KVM kernel module, QEMU guest implementation, libvirt virtualization management APIs and built-in error identification.

With an awareness of these components, the installation can configure an environment that supports high levels of hardware, firmware and functional redundancy, even when configured to run across multiple partitions (each hosting a distinct KVM hypervisor) in physical servers, remote sites, data mirroring with further automated recovery may be purchased and configured for the highest levels of HA. OpenStack can manage redundancy, spreading guest images across partitions.

Where is a good place to start?

What is involved in configuring a high availability environment? Before getting into the details, let’s consider the types of failures that we want to protect against. Examples include:

- System crash
- Network failures
- Storage issues

A simple starting point is a single primary host with a secondary backup, with one or more Linux guests. At a high level, such a deployment might look like the following:

- KVM for IBM z Systems running in two z Systems partitions in the same physical server, in an Active/Standby configuration. One KVM for IBM z image is configured as the primary partition. The other KVM for IBM z is your secondary “migration partner” to which the primary’s guests can be migrated if or when needed.
- Configure the network in each partition
    - Configure Linux bonding with redundant OSA ports, to protect against the loss of any individual adapter.
    - Install redundant load-balancing solution, such as IBM’s Websphere Edge Components, in the external network for requests to the host, to protect against loss of access to any individual guest or host. This way you’re ready to redirect work to your secondary guest or KVM partition when needed. Load Balancing solutions receive and distribute workload requests. Alternatively, you can configure the guests to work with virtual IP addresses and a dynamic network routing algorithm such as OSPF, which enables network packets to reach a guest via a single IP address through alternate paths in the case of a network issue.
- Configure storage access in each partition
    - SAN configuration: Configure two SAN fabrics each with disk storage systems with dual storage controllers. More information on this topic can be found in “SAN Fabric Administration Best Practices Guide, Support Perspective” (4).

    - Host Bus Adapter: Configure software- or firmware-based multi-pathing to enable transparent failover in the event of an adapter outage.
    - Ensure that the KVM host has access to primary and secondary disk systems
- Configure a cluster of Linux guests hosted by the KVMs for IBM z image
    - Use a guest clustering technology such as Websphere Application Server Network Deployment, which detects whether a node has failed, stops routing traffic to that node and routes it to surviving nodes for an active-active solution.
    - Define a copy of the KVM guest configuration on secondary storage, with access for failure recovery by a second KVM LPAR. This allows the guests to be restarted on the secondary KVM if the primary should fail.
- Configure a journaling file system, such as ext4, to ensure that you can quickly recover to a point of consistency in the wake of a system crash.

Now let's dive into these areas in greater detail.

KVM High Availability: Avoiding Single Points of Failure

As described earlier in this paper, KVM for IBM z Systems inherits the RAS capabilities offered by z Systems hardware and firmware. To ensure maximum availability for workloads in the environment, KVM for IBM z leverages Linux on z Systems resource management, including network adapater bonding, journaling file systems, and other Linux facilities. But the devil is in the details.

This section discusses the Linux, KVM and QEMU components involved in building a high availability environment, with a common goal of avoiding single points of failure.

The single point of failure considerations described in this section each relate to part of the stack described in Figure 1.

The Linux kernel:

The Linux kernel encompasses components and corresponding drivers for many of the hardware functions available in the firmware partition. Many of the components support critical host resources that each need to be configured with high availability in mind. Some of the components automatically support redundancy between Linux instances, and others need to be configured for redundancy. And a few remaining cases require mitigating actions to avoid availability exposures. The following describes the key host resources, along with their availability implications.

1. **Open Systems Adapter (OSA)**, the network adapter that supports Ethernet connectivity.

   a. OSA bonding provides redundancy for OSA ports on the same Ethernet LAN segment (layer 2) with failover that is transparent to the application. Use bonded network adapters to connect the KVM host to the outside world, while presenting a single virtual adapter to each guest, providing transparent adapter failover that is configured in one place for all guests.
   b. The KVM host loads the Linux “bonding” device driver, and uses it to enslave two OSA interfaces into a single, logical interface. Since OSA adapters are virtualized by the hardware, this means bonding two virtual OSA adapters that are located on two different physical OSA adapters. Doing so benefits the hypervisor and guests, and bonding automatically fails over connections when needed. Different bonding modes are supported, each with its own characteristics and implications (e.g., active/backup, load balancing)
2. **Virtual Switch** is software support that provides virtualized network connectivity to guests.
   a. Open vSwitch provides a switching stack while supporting multiple protocols and standards used in computer networks through standard management interfaces and protocols.
   b. MacVTap is a Linux device driver meant to simplify virtualized bridged networking. The macvtap endpoint can be used directly by kvm/qemu. It has its own mac address in the ethernet segment used to make both the guest and the host show up directly on the switch that the host is connected to.
   c. In both solutions, the guests’ use of virtual IP addresses (VIPA's) combined with a dynamic routing protocol such as Open Shortest Path First (OSPF) can dynamically select an alternate path to the destination IP address.
3. **Host Bus Adapter** has two flavors, for FCP and for FICON.
   a. With the FCP version, software-based multi-pathing support in Linux can provide transparent failover to an alternate FCP adapter port.
   b. With FICON, firmware-based multi-pathing support in the System z channel subsystem can provide transparent failover to an alternate FICON port without any software involvement.
   c. In both cases, the redundancy provided for storage controller adapters is conceptually similar to network bonding. With FCP, multi-pathing to the same device is handled by the KVM for IBM z host via the Linux kernel component dm_multipath (device mapper), which distributes I/O requests across paths. With FICON, hardware/firmware handles recovery, assuming a proper I/O configuration.
4. **Critical daemons:** With KVM for IBM z Systems, services are managed by the *systemd* service manager. With systemd, each service has its own configuration file with various parameters that can be specified, including under what conditions the service should be restarted (never, always, on failure, etc.). These values can be customized by the system administrator as desired. Example daemons include multipathd, libvirtd, hpmd, sshd, lvmd, clvmd, etc.

5. **Application & middleware** - Some middleware has its own built-in clustering and failover support. Examples include the Websphere application server (WAS) or the DB2 database environment. This support is often very granular and can take advantage of deep insight into the processing of that software. For example, you can take advantage of two-phase commit transactional recovery in Websphere, gaining data integrity benefits that you would not be able to achieve with an external cluster manager. Each approach is application-specific.
6. **SAN switches** enable fully redundant connections with dual fabrics. Data paths switch automatically when a failure is detected.
7. **Journal File system** facilitates tracking all interim changes to the file system so that they can be re-processed in the event of a failure. The file manager maintains a journal reflecting changes that were made. When restarting after a failure the file system replays the journal as needed to ensure the file system is consistent, without requiring the sometimes lengthy delays involved in running an out-of-band file system checker. Note that journal file systems are typically configured to keep track of only metadata, rather than the file data itself, to improve performance at the expense of potential data loss. Even when both metadata and file data are being journaled, there are still windows when data loss is possible.
8. **Mirrored storage subsystems** support data replication. Both ECKD and SCSI disk firmware offer synchronous and asynchronous disk mirroring that can be switched to the secondary storage in the event of a primary storage failure. Fixed Block storage supports mirroring between storage subsystems.

KVM / QEMU

KVM virtualizes system resources … CPU, memory, storage, network. The following areas are relevant for a High Availability configuration.

1. **Virtual networking**: Failure recovery is handled in the host (bonded host interface)
2. **Virtual storage:** Failure recovery is handled in the host, using FCP and/or FICON multi-pathing. A RAID (Redundant Array of Independent Disks) array within a storage controller can be used to protect against failure of individual disks.

QEMU supports network and storage emulation, providing accelerated I/O functions. The device emulation is provided via virtio interfaces.

1. **Device/processor virtualization:** Both ECKD and FCP devices are surfaced to KVM guests as virtio-blk devices; the host handles complexities such as multi-pathing and the guests inherit the benefits.
2. **Live guest relocation:** Used to avoid planned outages by moving one or more Linux guests from one KVM for IBM z hypervisor to another, across LPARs or physical servers.

3. **Restart a guest:** KVM guests can be automatically started by libvirt upon host IPL if desired. This can be done using the *virsh* command (“virsh autostart <guest>"), to enable autostart. Also, the libvirt-guests services enables the system administrator to automate certain actions against guests on hypervisor start and shut down. For example, all running guests can be suspended when the hypervisor stops and then resumed on start up.
4. **Secure remote management APIs:** Several choices exist for secure remote management configuration. Libvirt interfaces are used by solutions such as OpenStack. Other KVM open source utilities such as virt-tools and Virt-Manager are available but not delivered with KVM on IBM z Systems.
5. **Command line utilities:** the *virsh* command can start and stop KVM guests, and provides a wide variety of other query and update functions used to maintain guest configuration.

Linux Guest High Availability

The majority of this paper is focused on the KVM hypervisor and how to configure for high availability. But what about the Linux guest virtual machines? Some guest availability is provided seamlessly by KVM.

However, coarse level failure detection is usually not good enough. Your guest operating systems are likely performing critical business applications, where high availability considerations such as data replication, function recovery and database integrity are all critical aspects of supporting a 24x365 level of service availability. Linux guest high availability is often dependent on cluster-aware middleware that is deployed to manage processing that is replicated across multiple hypervisors. Middleware such as Websphere and DB2 HADR each manage workload, and in the event of a failure ensure that processing continues at an alternate guest, the goal of failover. Cluster managers support active/active and active/standby architectures, with differing levels of cost to ensure that your workload remains available even though its primary system may have failed.

Standard practices such as implementing an external load balancer to distribute work and avoid sick or dead systems, are often required.

Guest availability must also take planned change into account, using live migration to move work to another hypervisor before a planned host outage is initiated.

KVM Configuration Scenarios

In the next few sections, we show sample configuration scenarios. Each scenario delivers a set of strengths related to high availability.

Single KVM Host

The single KVM host environment has been discussed in detail in previous sections. It is a good starting point, supporting restart in place for failed guests, while exploiting TCP/IP, data and application takeover with VIPA. Data may be mirrored using hardware-driven mirroring, such as SAN Volume Controller (SVC), which handles mirroring at the disk controller.

In addition, active-active KVM guest mirroring can be provided using Websphere Application Server Network Deployment

Multiple KVM hosts

With the scenario shown in Figure 2, we deploy at least two KVM host environments on one or more servers. In either case, we must carefully allocate sharable resources to the host partitions. HA configuration requirements are essentially the same regardless of the number of CPCs. Multi-pathing is configured to cross adapter boundaries for improved redundancy. The single server configuration is a good initial option since the z Systems hardware is unlikely to fail. However, many businesses require the additional protection obtained by spreading the workload across two physical servers. The obvious value of these scenarios is that the primary KVM can failover to the secondary KVM in the event of a KVM failure.

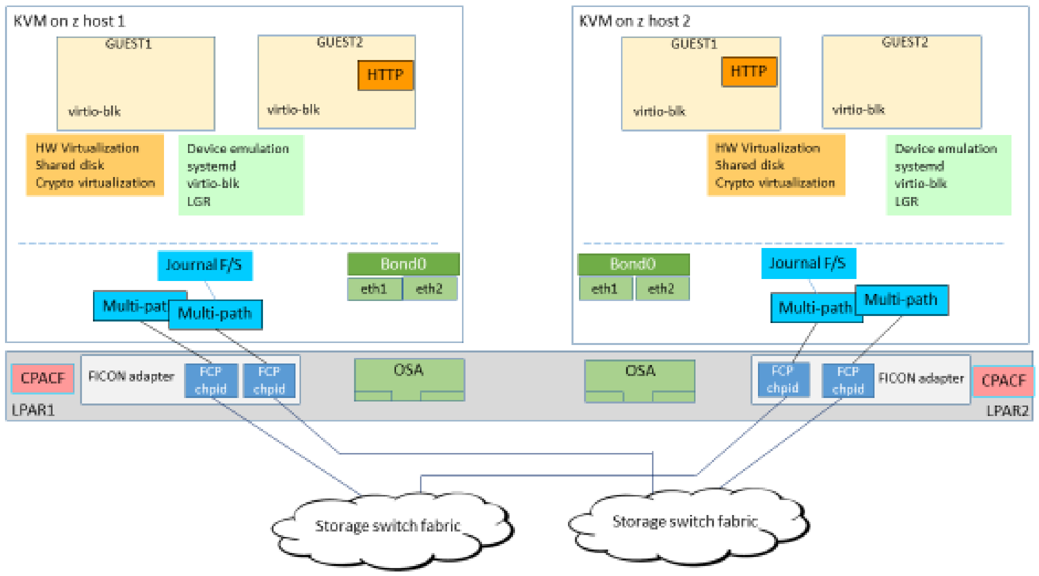

Figure 2: Configuring multiple KVM hosts

Just a reminder: Guest virtual machines do not have passthrough access to devices. Only virtio-blk access is provided for KVM guests, with the disk attached to the hypervisor.

Hardware outages still happen, however infrequently, and hardware maintenance could introduce availability windows as well. To eliminate the resulting availability risk, businesses that require a higher level of available should spread processing over two or more servers. This is also the basis for configuring a disaster recovery scenario when the alternate server is at a remote site.

Disaster Recovery (DR)

Disaster Recovery has been a focus of IT planning for many years, with varying offerings and approaches available to accomplish DR. Several options rely on offsite copies of business data, but have high recovery times and levels of data loss. As a result automated disk replication should be planned for the highest levels of availability. This includes options such as

- Active storage replication to a remote site, without recovery automation, with seconds to minutes of data at risk of needing to be recreated, and hours or days until data can be fully recovered after a failure.
- Active storage replication to a remote "in-house" site, with zero to minutes data exposure (depending on replication technology and automation policy), and one or more hours recovery time (dependent on automation)
- Active software replication to a remote "active" site, with seconds to minutes of data risk, and seconds to minutes until recovery has been completed.

Due to the cost of an outage to many businesses, most look for greater levels of availability, covering a wider range of events and scenarios.

The following are disaster recovery considerations for KVM for IBM z Systems:

- What level of D/R is commensurate with the level of investment
    - Data mirroring is available in basic KVM for IBM z configuration, but with manual intervention to handle each of the steps. Mirroring can be set up using SAN Volume Controller (SVC) or DS8000 disk. A manual procedure would need to be followed to shut down KVM for IBM z in one partition, failover the mirrored disks to a second partition, and manually start up KVM for IBM z in the second partition. Note that you can obtain a SVC and back it with your choice of supported disk. SVC is also included in products like the IBM Storwize® V7000, or IBM FlashSystem® V840 or V9000 storage solutions.

    - Active/standby or Active/Active configuration works for basic disaster recovery across multiple partitions, such as when using the Websphere Application Server Network Deployment
- Mirroring techniques
    - Mirror disk by disk: KVM virtualizes all disks to its guest operating systems. As a result, guests have no information about the physical storage topology. Therefore, all KVM disks used for guest data should be

mirrored together, through a "consistency group", to maintain data consistency. A consistency group is a method of replicating a group of physical disks and maintaining data consistency across all disks in the group as well as keeping the replication direction the same for all disks in the group.

Summary

KVM for IBM z Systems provides open source virtualization for IBM z Systems and the LinuxONE platforms. Using the combination of KVM virtualization and IBM z Systems and LinuxONE, different Linux workloads are supported with the high levels of RAS offered by z Systems hardware. With an understanding of the base RAS highlights, and knowledge of the KVM and Linux environment, you can create a system that avoids single points of failure and provides very high levels of availability in support of Linux business applications, hosting a cloud environment, or integrating with a distributed server farm. This paper has highlighted considerations in building such an environment. This information, in turn, allows you to begin to design a high availability environment that meets your business's requirements.

References

1. *High Availability Architectures for Linux on IBM System z,* by Steve Weir, Scott Loveland, Harriet Morrill (June 15, 2010)
2. *High Availability of System Resources: Architectures for Linux on IBM System z Servers*, by Scott Loveland (September, 2012); http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=SA&subtype=WH&htmlfid=ZSW03236USEN
3. *IBM GDPS Family of Products, An Introduction to Concepts and Capabilities, an IBM Redbook*, http://www.redbooks.ibm.com/redbooks/pdfs/sg246374.pdf
4. *SAN Fabric Administration Best Practices Guide, Support Perspective, by Brocade,* http://www.brocade.com/content/dam/common/documents/content-types/whitepaper/san-admin-best-practices-bp.pdf
5. "KVM virtualization and management tutorial", http://searchservervirtualization.techtarget.com/KVM-virtualization-and-management-tutorial
6. Linux on System z: Reliability, Servicability, and Availability, by Dr. Reinhard Buendgen, SHARE March 16, 2012

Acknowledgements

- Tony Gargya (KVM for IBM z architect), for his thorough assistance with KVM for IBM z specifics
- William Clarke (z Systems RAS Engineering), for his insight on z Systems hardware RAS