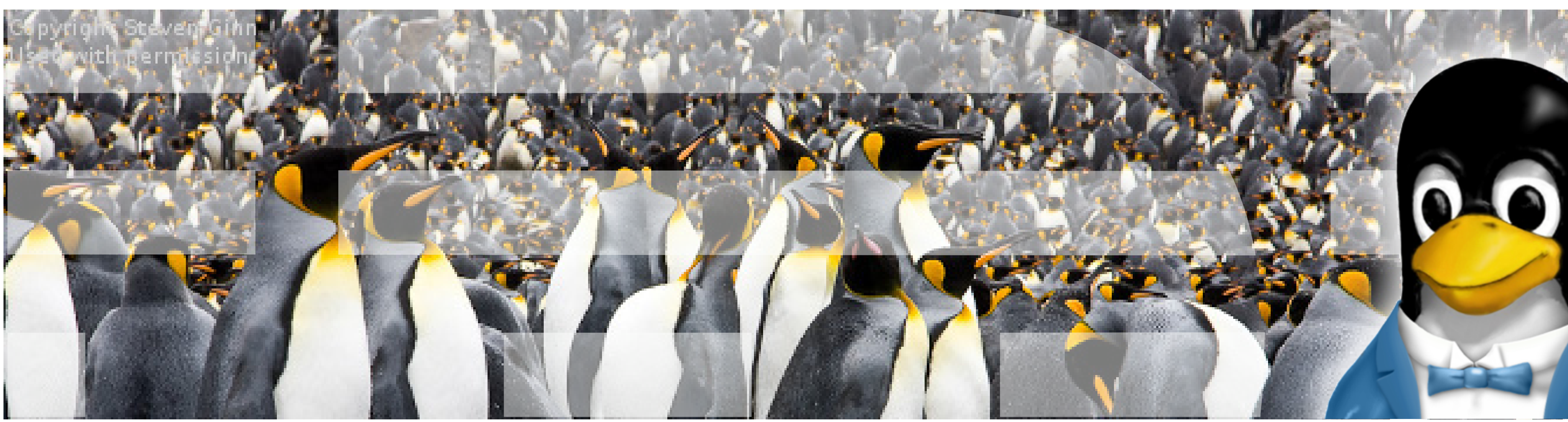
Resource Over-Commitment for KVM Virtualization Environments

Karl Rister
IBM Corporation
Linux Technology Center



Linux Foundation Collaboration Summit
April 4, 2012
San Francisco, CA, USA

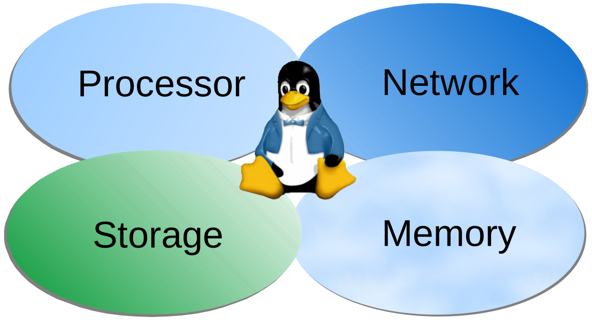
Virtualization over-commitment overview

What does it mean to over-commit resources in (KVM) virtualization?

- Virtualization over-commitment is using the virtualization technology (hypervisor) to allocate more of a resource to guest virtual machines than is present in the system
- Virtualization over-commit capabilities have many parallels in traditional operating system concepts
- The amount of similarity between virtualization over-commitment and traditional operating system over-commitment is dependent on the type of resource being over-committed
- KVM is Linux + hypervisor technology

Virtualization over-commitment overview

- Processor over-commitment
  - Creating more virtual processors (VCPU) for guest virtual machines than physical processors available
  - Example: 32 x 1 VCPU guest virtual machines running on a 4 CPU core system
- Memory over-commitment
  - Allocating more memory to guest virtual machines than physical memory available
  - Example: 32 x 1 GB guest virtual machines running on a 16 GB system
- Storage over-commitment
  - Allocating more storage capacity to guest virtual machines than is available
  - Example: 32 x 100 GB guest virtual machine virtual disks on a 1 TB hard drive
- Network over-commitment
  - Sharing a physical network connection amongst more than 1 guest virtual machine and/or the hypervisor
  - Example: 32 guest virtual machines each with a virtual network adapter sharing a 1 Gb network connection



Resource contention

What happens if not enough resources exist?

- Depends on the resource in question
  - Processor
    - Poor performance
    - Variable performance
  - Memory
    - Poor performance
    - Variable performance
    - Allocation failures
    - OOM
    - Data corruption
  - Storage
    - `-ENOSPC`
    - Guest virtual machines paused
    - Data corruption
  - Network
    - Poor performance
    - Variable performance
- Impact

Resource contention

- What can lead to resource contention?
  - Overly aggressive over-commitment policy
  - “Thundering herd”
  - “Run on the bank”
  - “Noisy Neighbor”

- Contention resolution
  - Any over-commitment policy should include a plan for resolving resource contention issues
  - Mitigation techniques
    - Migration
    - Pausing
    - Destroying

KVM processor over-commitment

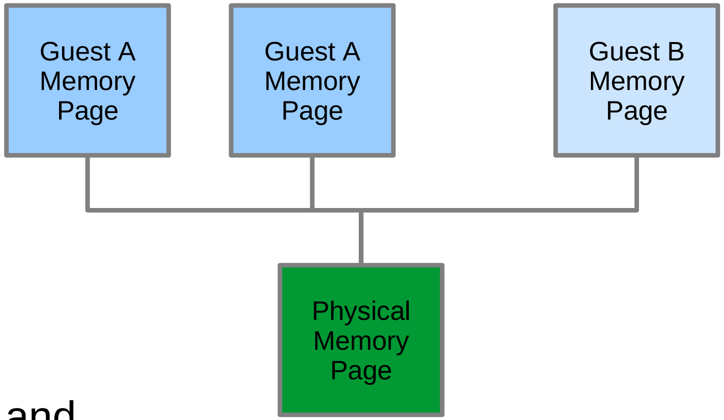
How does KVM do processor over-commitment?

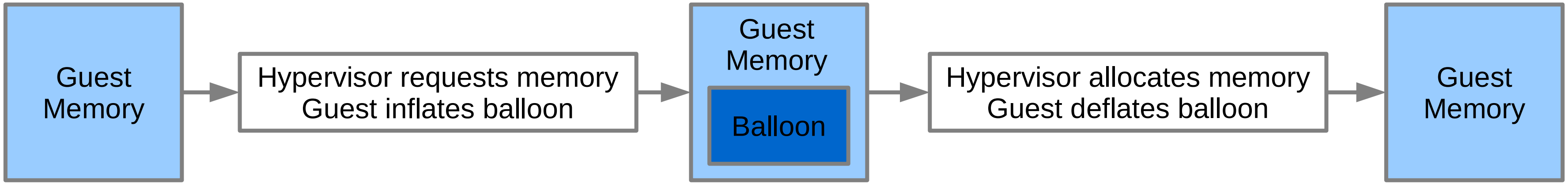
- Easily
- Simplest form of over-commitment
- Risk Level : Low
- Performance implications
- Problem resolution

KVM memory over-commitment

How does KVM do memory over-commitment?

- Three technologies support KVM memory over-commitment
  - Paging (swapping)
    - Least performant
    - Highest risk
    - Managed by Linux kernel
  - Page sharing
    - KSM - Kernel Shared Memory
    - A guest can share memory with itself
    - Aided by intelligent userspace management
  - Ballooning
    - Cooperative mechanism between hypervisor and guest virtual machine
    - Requires guest virtual machine reciprocity
    - Requires intelligent userspace management

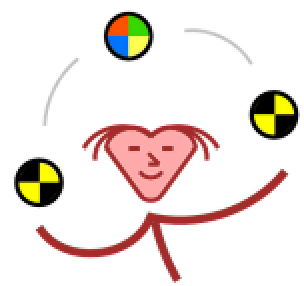




KVM memory over-commitment

- Memory management
  - Linux controls paging
  - KSM aided by management
  - Ballooning requires management

- MOM – Memory Overcommitment Manager
  - Management for Page Sharing and Ballooning
  - Policy driven
  - Extensible
  - Dependent on guest virtual machine cooperation



KVM memory over-commitment

- Risk level : High
  - Negative technology interactions
  - Possible scenarios
    - Swap storm
    - Allocation failure
      - Inside the guest virtual machine
      - Inside the hypervisor
  - Mitigation strategies are required

- Problem resolution

Storage over-commitment

How does KVM do storage over-commitment?

- Special Files / File formats
  - Sparse files
    - Raw format
    - Management issues
  - QCOW2
    - Special file format that enables many compelling features
      - Encryption
      - Compression
      - Snapshots
      - Over-commit
    - Accurately reports correct size

Storage over-commitment

```
[root@host images]# qemu-img create sparse.img 100G
Formatting 'sparse.img', fmt=raw size=107374182400

[root@host images]# qemu-img create -f qcow2 qcow2.img 100G
Formatting 'qcow2.img', fmt=qcow2 size=107374182400 encryption=off
cluster_size=65536

[root@host images]# ls -lash *.img
136K -rw-r--r--. 1 root root 256K Mar 26 14:01 qcow2.img
   0 -rw-r--r--. 1 root root 100G Mar 26 14:00 sparse.img
```

Storage over-commitment

- Other solutions
  - Unlike previous over-commit topics (processor, memory) storage can be over-committed using “external” capabilities
  - No requirement to depend on Linux/KVM specific features
  - Storage servers
    - Snapshots
    - Thin provisioning
    - Deduplification
  - Relocate management overhead to a different service tier

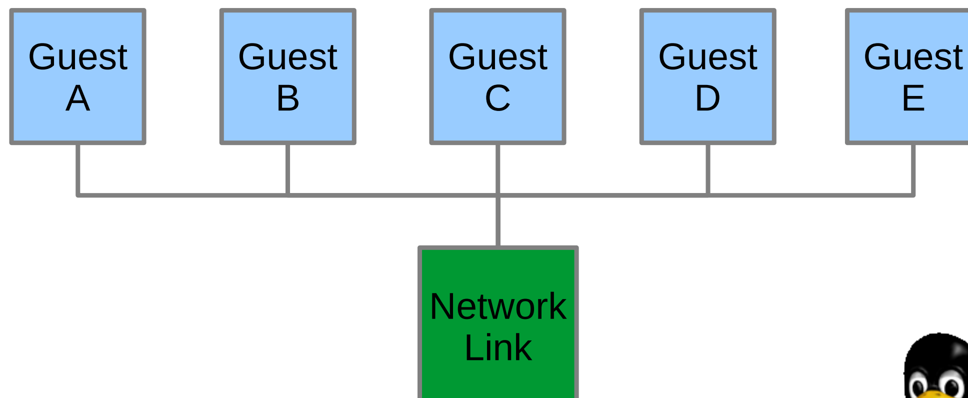
- Risk level : Medium
  - Corruption can occur in the absence of proper recovery
  - Mitigation strategies required

- Problem resolution

Network over-commitment

How does KVM do network over-commitment?

- Similar to processor over-commitment
- Each guest virtual machine looks and acts like a process sharing a network connection
- Available bandwidth shared
- Risk level : Low
- Problem resolution



Discussion

Questions?

Additional resources

Links to additional information:

- KVM : http://www.linux-kvm.org/page/Main_Page
- QEMU : http://wiki.qemu.org/Main_Page
- libvirt : http://libvirt.org/
- MOM : https://github.com/aglitke/mom/wiki

Legal

Trademarks and Disclaimers

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries. For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml:

IBM, the IBM logo, BladeCenter, Calibrated Vectored Cooling, ClusterProven, Cool Blue, POWER, PowerExecutive, Predictive Failure Analysis, ServerProven, System p, System Storage, System x , System z, WebSphere, DB2 and Tivoli are trademarks of IBM Corporation in the United States and/or other countries. For a list of additional IBM trademarks, please see http://ibm.com/legal/copytrade.shtml.

The following are trademarks or registered trademarks of other companies:

Java and all Java based trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries or both
Microsoft, Windows,Windows NT and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
UNIX is a registered trademark of The Open Group in the United States and other countries or both.
Linux is a trademark of Linus Torvalds in the United States, other countries, or both.
Cell Broadband Engine is a trademark of Sony Computer Entertainment Inc.
InfiniBand is a trademark of the InfiniBand Trade Association.

Other company, product, or service names may be trademarks or service marks of others.

NOTES:
Linux penguin image courtesy of Larry Ewing (lewing@isc.tamu.edu) and The GIMP

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software design and configuration. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Users of this document should verify the applicable data for their specific environment.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

Information is provided “AS IS” without warranty of any kind.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

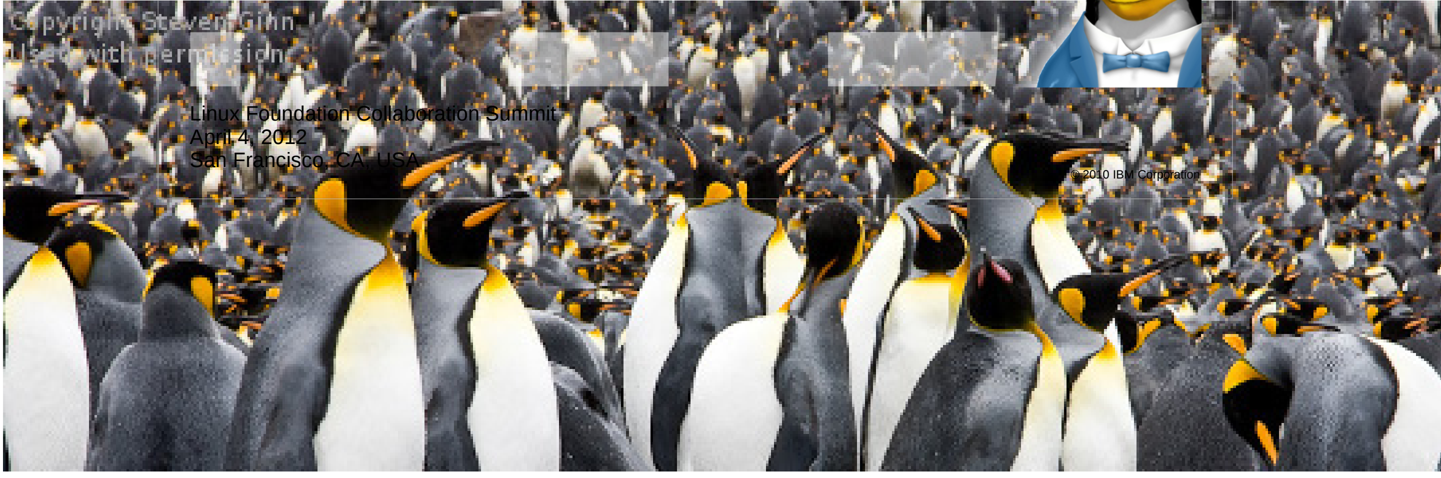
Prices are suggested US list prices and are subject to change without notice. Starting price may not include a hard drive, operating system or other features. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Any proposed use of claims in this presentation outside of the United States must be reviewed by local IBM country counsel prior to such use.

The information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any

Resource Over-Commitment for KVM Virtualization Environments

Karl Rister
IBM Corporation
Linux Technology Center



Linux Foundation Collaboration Summit
April 4, 2012
San Francisco, CA, USA

Virtualization over-commitment overview

What does it mean to over-commit resources in (KVM) virtualization?

- Virtualization over-commitment is using the virtualization technology (hypervisor) to allocate more of a resource to guest virtual machines than is present in the system
- Virtualization over-commit capabilities have many parallels in traditional operating system concepts
- The amount of similarity between virtualization over-commitment and traditional operating system over-commitment is dependent on the type of resource being over-committed
- KVM is Linux + hypervisor technology

Virtualization / Operating system parallels

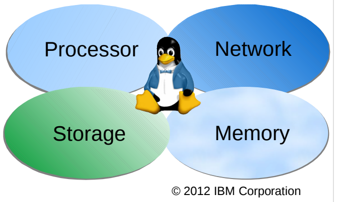
- Virtual memory
- Multitasking

KVM is Linux

- Guest virtual machine is a standard Linux process
- Over-commit in KVM is just like over-commit in Linux
- Interesting interactions and secondary effects do exist

Virtualization over-commitment overview

- Processor over-commitment
  - Creating more virtual processors (VCPU) for guest virtual machines than physical processors available
  - Example: 32 x 1 VCPU guest virtual machines running on a 4 CPU core system
- Memory over-commitment
  - Allocating more memory to guest virtual machines than physical memory available
  - Example: 32 x 1 GB guest virtual machines running on a 16 GB system
- Storage over-commitment
  - Allocating more storage capacity to guest virtual machines than is available
  - Example: 32 x 100 GB guest virtual machine virtual disks on a 1 TB hard drive
- Network over-commitment
  - Sharing a physical network connection amongst more than 1 guest virtual machine and/or the hypervisor
  - Example: 32 guest virtual machines each with a virtual network adapter sharing a 1 Gb network connection



Processor over-commitment

Similar in concept to operating system's multitasking capabilities

VCPU scheduling managed by the hypervisor just as an operating system schedules processes

Memory over-commitment

Similar in concept to operating system's virtual memory capabilities

Memory management provided by the hypervisor

Some overlap with operating system design

Potential for virtualization specific optimizations

Storage over-commitment

Can be managed by the hypervisor or by an external storage controller

Little hypervisor management until storage usage nears (or achieves) full capacity

Network over-commitment

Similar in concept to traditional networking where multiple processes on a system all utilize the same physical network connection

Also similar to many systems connected to a switch sharing a single uplink connection

Resource contention

What happens if not enough resources exist?

- Depends on the resource in question
  - Processor
    - Poor performance
    - Variable performance
  - Memory
    - Poor performance
    - Variable performance
    - Allocation failures
    - OOM
    - Data corruption
  - Storage
    - `-ENOSPC`
    - Guest virtual machines paused
    - Data corruption
  - Network
    - Poor performance
    - Variable performance
- Impact

Impact

Understood that failure is undesirable, but what about the gray areas between failure and everything works?

Poor/variable performance

Depends on desired goal:

Best performance?

Best density?

Consistent performance?

Resource contention

- What can lead to resource contention?
  - Overly aggressive over-commitment policy
  - “Thundering herd”
  - “Run on the bank”
  - “Noisy Neighbor”
- Contention resolution
  - Any over-commitment policy should include a plan for resolving resource contention issues
  - Mitigation techniques
    - Migration
    - Pausing
    - Destroying

Resource contention

It is possible to be too aggressive AND it is possible to just get unlucky

Discuss scenarios as each topic is covered

KVM processor over-commitment

How does KVM do processor over-commitment?

- Easily
- Simplest form of over-commitment
- Risk Level : Low
- Performance implications
- Problem resolution

How

KVM guest virtual machines are Linux processes

Each VCPU is a thread within that process

Linux supports multitasking via a highly optimized task scheduler (CFS)

Risk Level

Long History of multitasking in Linux

There are potential performance implications

Performance implications

Increased scheduler overhead

Processor resource contention

Variable performance

Problem resolution

Resource controls

CPU shares

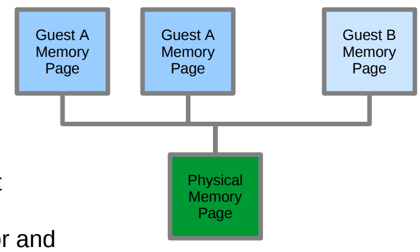
VCPU capping

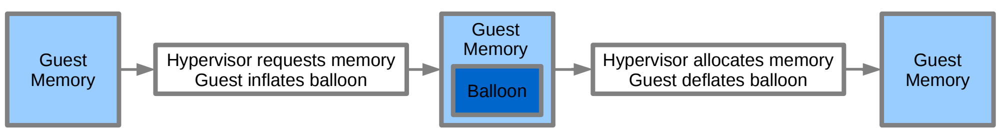
Mitigation (migration/pausing/destroying)

KVM memory over-commitment

How does KVM do memory over-commitment?

- Three technologies support KVM memory over-commitment
  - Paging (swapping)
    - Least performant
    - Highest risk
    - Managed by Linux kernel
  - Page sharing
    - KSM - Kernel Shared Memory
    - A guest can share memory with itself
    - Aided by intelligent userspace management
  - Ballooning
    - Cooperative mechanism between hypervisor and guest virtual machine
    - Requires guest virtual machine reciprocity
    - Requires intelligent userspace management





Paging (swapping)

Standard component of Linux virtual memory subsystem

Use storage space as supplemental memory capacity

Guest virtual machine unaware of hypervisor actions

Large performance impact

Increased code path length

Storage is slow

Linux LRU / Intel VT interaction

Page sharing

Sometimes also called Kernel Samepage Merging

Identifies candidate pages via scanning thread

Pages shared if contents identical

Copy-on-Write used if page is modified

Guest virtual machine unaware

Minimal performance impact

CPU cycles

NUMA locality

THP

Ballooning

Free memory with hypervisor requests

Allocate memory with hypervisor grant

Minimal performance impact

Temporarily reduces guest memory size

Guest virtual machine paging

Can increase I/O (paging, less cache)

THP

KVM memory over-commitment

- Memory management
  - Linux controls paging
  - KSM aided by management
  - Ballooning requires management

- MOM – Memory Overcommitment Manager
  - Management for Page Sharing and Ballooning
  - Policy driven
  - Extensible
  - Dependent on guest virtual machine cooperation

Memory management

Linux paging decisions affected by overall memory footprint

Page/buffer cache can influence decisions

KSM can be statically configured via sysfs (like proc) but dynamic control is optimal

Ballooning requires explicit commands

Something must decide whether to inflate or deflate the balloon and by how much

MOM

IBM initiated open source memory over-commitment management tool

Currently included in Fedora 16

Pushing for inclusion in enterprise distributions

Continuously collects data from hypervisor and guest virtual machines

Uses data as input to a decision engine

Default policy

Acts when hypervisor is under memory pressure

Tunes KSM

Initiates ballooning operations

Aggressiveness reflects degree of pressure

Strives to avoid paging

Administrators can write their own policy to target different goals

KVM memory over-commitment

- Risk level : High
  - Negative technology interactions
  - Possible scenarios
    - Swap storm
    - Allocation failure
      - Inside the guest virtual machine
      - Inside the hypervisor
  - Mitigation strategies are required

- Problem resolution

Risk level

Memory over-commit failure scenarios difficult to recover from (if at all)

Technology interactions (previously mentioned)

Indirectly causes CPU resource contention

KSM vs. THP

THP vs. Ballooning

Increased pressure on I/O subsystem

Optimizations compete

Consolidate (provision) like guests on a single system to optimize page sharing

What happens when they all need memory at the same time?

Problem resolution

Comprehensive memory management required

Difficult to independently manage the technologies and their interactions

MOM

Mitigation (migration/pausing/destroying)

Storage over-commitment

How does KVM do storage over-commitment?

- Special Files / File formats
  - Sparse files
    - Raw format
    - Management issues
  - QCOW2
    - Special file format that enables many compelling features
      - Encryption
      - Compression
      - Snapshots
      - Over-commit
    - Accurately reports correct size

Sparse files

Special files that only allocate used blocks

File system feature

Raw format

No special capabilities

Cannot leverage snapshots (image reuse)

Management issues

File system reports full size (not used size)

QCOW2

Features add over-head

Increased allocation code paths

Block re-mapping introduces randomness

Increases pressure on I/O subsystem

Storage over-commitment

```
[root@host images]# qemu-img create sparse.img 100G
Formatting 'sparse.img', fmt=raw size=107374182400

[root@host images]# qemu-img create -f qcow2 qcow2.img 100G
Formatting 'qcow2.img', fmt=qcow2 size=107374182400 encryption=off
cluster_size=65536

[root@host images]# ls -lash *.img
136K -rw-r--r--. 1 root root 256K Mar 26 14:01 qcow2.img
   0 -rw-r--r--. 1 root root 100G Mar 26 14:00 sparse.img
```


Image file creation

- Simple command invocations
- Likely abtracted by management infrastructure such as libvirt

File differences

- Special “ls” arguments show blocks allocated in addition to file size
- Empty sparse file appears to be 100 GB but uses 0 blocks
  - Not obvious with traditional query tools
- Empty QCOW2 file is 256 KB but uses 136 KB already
  - File format overhead evident
  - Easily determine actual allocation

Storage over-commitment

- Other solutions
  - Unlike previous over-commit topics (processor, memory) storage can be over-committed using “external” capabilities
  - No requirement to depend on Linux/KVM specific features
  - Storage servers
    - Snapshots
    - Thin provisioning
    - Deduplication
  - Relocate management overhead to a different service tier

- Risk level : Medium
  - Corruption can occur in the absence of proper recovery
  - Mitigation strategies required

- Problem resolution

Other solutions

No requirement to use/depend on Linux/KVM specific features

Reduced management overhead

Single/few resources to monitor

Increase sharing savings

Snapshots

Deduplication

Can be combined with Linux/KVM specific capabilities

Risk level

Guest virtual machines pause on write failure

Administrator required to recover and resume guest virtual machine

Problem resolution

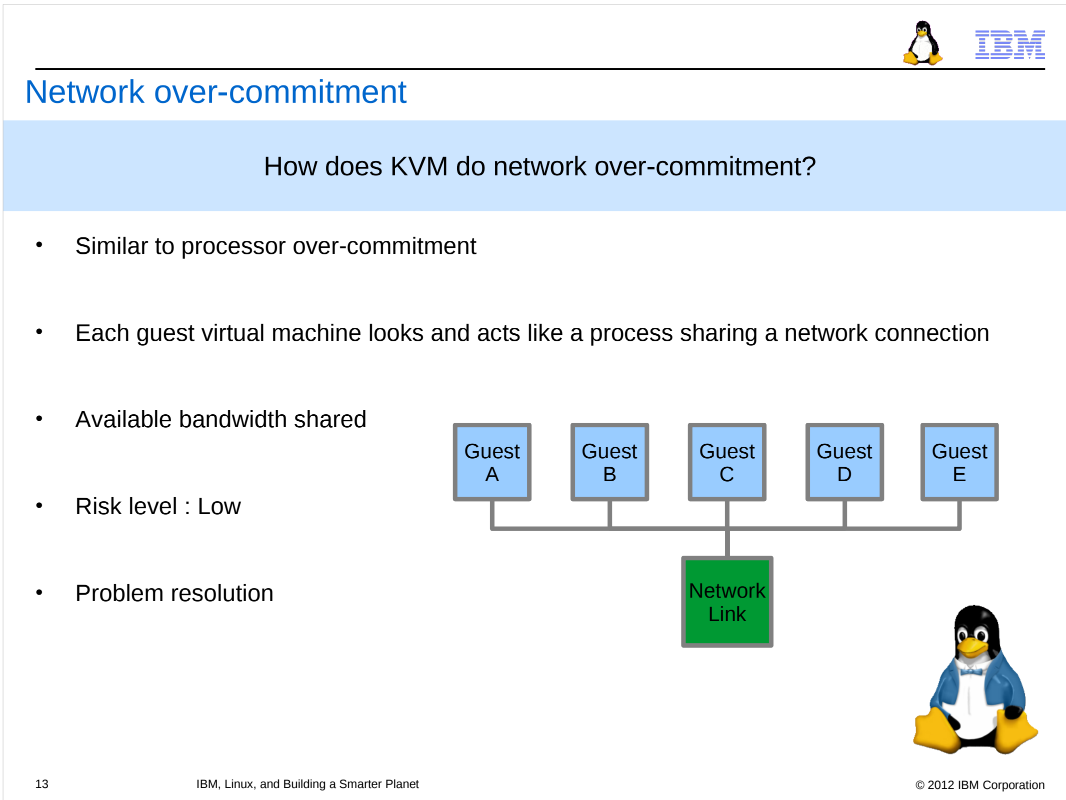
Usually requires administrative intervention

Add storage capacity

Reduce existing storage footprint

Mitigation (migration/pausing/destroying)

Migration requires advanced solution planning



Available bandwidth shared

- Default distribution determined by efficiency
  - Theoretically equal
  - Guest virtual machines with highly efficient traffic pattern (large packets) will probably achieve better throughput
- Distribution can be controlled via resource controls (cgroups)

Problem resolution

- Lack of problems limits solutions
- Mitigation (migration/pausing/destroying)

Discussion

Questions?

Additional resources

Links to additional information:

- KVM : http://www.linux-kvm.org/page/Main_Page
- QEMU : http://wiki.qemu.org/Main_Page
- libvirt : http://libvirt.org/
- MOM : https://github.com/aglitke/mom/wiki