VMware, Inc. Invention Disclosure Form

Overview

File #:

Title*

A METHOD TO OPTIMIZE MEMORY PAGE TRANSPORTATION IN VMOTION

Technology Group:

Technology Group Category*

SDDC Compute

Product or Technology*

SDDC Compute - Memory

Received Date

vmware

VMware, Inc. Invention Disclosure Form

Disclosure

Previous Public Disclosure:

Has this invention been made known to anyone outside of VMware and not subject to an NDA?*

No

When?

How?

Anticipated Public Disclosure:

Outside of a non-disclosure agreement (NDA), is there any planned disclosure of this invention or release of a product incorporating this invention to anyone outside of VMware?

No

When?

How?

Supporting Documents:

Documents and Attachments:	
Document Name	**Subject**

vmware

VMware, Inc. Invention Disclosure Form

Third Party Interest

Development Funding:

Is the development of this invention being funded by an agency of the U.S. Government?*

No

Agency

Contract Number

Special Contract Limitations:

Are you under a duty to maintain any aspect of this invention in confidence or to assign all or any part of it to any other individual (for example, business partner), company (for example, previous employer), organization, or educational institution?*

No

Describe

Have you developed all or any part of this invention using the equipment, consulting services, or facilities of any other individual, company, organization or educational institution?*

No

Describe

Do you know of any other potential limitations to VMware's exclusive right to own or develop this invention?*

No

Describe

VMware Project:

Is the development of this invention related to a current VMware Project?*

Yes

Enter Name (or Codename) of Project:*

vsphrere

Is this a project that VMware is collaborating on with a third party (or parties)?*

No

Enter Name of Third Parties:

Supporting Documents:

Documents and Attachments	
Document Name	**Subject**

vmware

VMware, Inc. Invention Disclosure Form

Invention Description

Problem to be Solved:

What problem does the invention solve?*

In current ESXi, during VMotion source host needs to send the non-zero page to destination host when detecting zero page configuration is enabled(MIGRATE_DETECT_ZERO_PAGES). For present ESXi design and implementation, when source host detects a non-zero page, it will copy one complete memory page with 4K bytes to network socket buffer then sends non-zero pages to the destination host.At destination, a new memory page is allocated and data is copied from socket buffer received from source ESXi host. So when virtual machine has large vRAM and non-zero memory pages account for quite proportion among whole memory pages, vMotion may consume pretty much network bandwidth to transfer amounts of non-zero pages.

Summary of the Invention:

Briefly, summarize the invention and how it solves the stated problem.*

During vMotion, virtual machine memory is transferred from source ESXi host to destination ESXi host via network, the current method of memory page transmission is to copy the whole non-zero page into socket buffer then send it out, the destination ESXi host receives this buffer, then allocates a new page and copies data from socket buffer to this new page.
We did an investigation on non-zero memory pages data for kinds of VMs in ESXi. We finds a common conclusion of non-zero memory pages layout. For example per Windows 7 and RHEL6, on the average the continuous 0 bits at the start and the end of page may cover up to 6~7% of a page on Windows 7 and 8~9% of a page on RHEL6, the percentage of continous zero bits at the start and end of a page can be higher if we run certain applications on operating system for example apache server.

Based on memory non-zero pages layout of common VM, our method is to improve the VM memory transmission by reducing the amount of data transferred via network, with our new method the average bandwdith saving could be cut down 8% for vMotion.

Detailed steps are as following.
Each memory page size of VM is 4096 bytes, we divided it to 64 blocks and then each block has 64 bytes. The block is minimum checking unit. If we find there are full zero bits block at the start or the end of a page, it will be stripped, and then only non-zero blocks in the middle of the page could be transferred to destination host.

ESXi Source Host:
1. The 64 byte block is checked whether it is a complete zero bits block, if yes, redo step 1 with the next block until the ending block of this page.If the block is no-zero, go to step 2.
2. Get the start position of non-zero block, if it’s not the ending of current page, go to step 3.
3. Go to the ending block of this page and checking if it's zero bits filled block, if yes, redo the steps 3 with

previousblock until reaching the start position got in step 2, if no go to step 4.
4. Get the end position of non-zero block with in current page.
5. Copy the start position and end position to meta-data of this page.
6. Copy the data between start and end position in page to page metadata in socket buffer.
7. ESXi source host send the buffer data to destination host.
ESXi Destination Host:
1. ESXi destination received buffer data from source host.
2. Extract the start and end position from meta data of current page.
3. Allocate a new page for destination virtual machine.
4. Fill the memory page with zero from 0 to start position.
5. Fill the memory page with zero from end position to end of page (PAGE_SIZE is 4096 bytes).
6. Copy the socket buffer data to new page between start and end position and the copy length is (end position - start position).

This method will not introduce any extra effort compared with ESXi current design, because with configuration MIGRATE_DETECT_ZERO_PAGES enabled source ESXi host will always check if a VM page is zero or not, this behavior have the same workload as our method in step 1, the start and end position information of page in step 6 are also stored in unused space in page metadata, no extra data structure is needed. With plenty rounds of experiments, by applying the method above we will strip the zero bytes at the start and the end of page and averagely this can save 6~9% vMotion transmission bandwidth.

Prior Art:

How have others tried to solve the problem and in what ways have their solutions been inadequate?

Supporting Documents:

Documents and Attachments	
Document Name	**Subject**
vmotionzerostripped.pdf	VMotionSteps

vmware

VMware, Inc. Invention Disclosure Form

Inventors

Inventor [1]	
Full Name:	Xie, Hongsheng
Home Address:	, US
Work Email:	hxie@vmware.com
Citizenship:	N/A
Office Location:	Level 8, South Wing of Tower C Raycom Info Tech Park No. 2 Kexueyuan South Road, Haidian District Beijing, 100190
Work Phone Number:	650-427-5000

vmware

VMware, Inc. Invention Disclosure Form

Inventors

Inventor [2]	
Full Name:	Liu, GengSheng
Home Address:	
Work Email:	gengshengl@vmware.com
Citizenship:	N/A
Office Location:	3401 Hillview Avenue Palo Alto, CA 94304
Work Phone Number:	650-427-5000

VMware, Inc. Invention Disclosure Form

Inventors

Inventor [3]	
Full Name:	Wang, Hao
Home Address:	, CN
Work Email:	haowang@vmware.com
Citizenship:	China
Office Location:	2 Kexueyuan South Road Haidian District, Beijing, China. 8Floor, Raycom InfoTech Park Tower C Beijing, CN 100190
Work Phone Number:	86-010-599-34234

vmware

VMware, Inc. Invention Disclosure Form

Inventors

Inventor [4]	
Full Name:	Zhao, Yimin (Hill)
Home Address:	, US
Work Email:	hillzhao@vmware.com
Citizenship:	China
Office Location:	8th Floor South Wing Tower C, Raycom InfoTech Park No. 2 Kexueyuan South Road Haidian District Beijing Beijing, Beijing 100190
Work Phone Number:	8601058746639