vMotion Streams

Arun
VMWare

Streams Overview

- Single, unified means for all VMotion consumers to send data
- Provides a flexible interface for sending large amount of data
- Efficiently transmit data across the network
- Transparent use of multiple threads and sockets

Architecture

- Streams are divided into two sides
- The generator
  - Send side
  - generator is the host pushing data into the stream.
- The handler
  - Recv side
  - The handler is the host reading the data from the network and processing it.
- Both source and destination host can fill either role.

Generator

- Supply data for the stream core to send over the network.

- Two stream variables:
  - dataCount - captures the amount of data available for sending.
  - streamData - points to data owned by the stream core.

- Stream core takes ownership of the streamData memory space
- Freed upon stream termination.

Handler

- Read-side
- Handle the consumer-level header
- Loop
  - Stream core calls completion functions until data is exhausted
  - prepare completion function
  - read completion function
- The consumer prepares a list of buffers for incoming data
  - Buffers: [address (VA), length] pairs
- Prepare completion: Populating the a set of VAs
- Read completion function: Finalize the set of VAs.

Stream Core

- Execute stream functionality
- Loop
  - Checks for any work
  - Runs through the generator routines
  - Runs through the handlers routines
  - If no work sends keep alive

Stream Helpers

- vmkernel system world
- Executes stream core code
- By default there are two helpers worlds
- Each helper world open a tcp sockets and binds it to a vmkernel nic
- So we have two separate TCP connections to the remote host

Stream Types

- Streams can either be synchronous or asynchronous.

- Synchronous Streams
  - Stream Helpers execute until the generator function changes dataCount to 0.
  - Page In

- Asynchronous Streams
  - Stream helper threads will run continually so long as dataCount non-zero, or work is available.
  - The moment dataCount reaches 0, the stream core will handle other consumers until more data becomes available.

Stream Synchronization

- StreamIdleBarrier can be used to impose synchronization points between both hosts.
- StreamIdleBarrier will block until both hosts have completed processing of all pending and outstanding stream data.

- Hard barrier
  - Stop transmission on the send side
  - End of precopy

- Soft barrier
  - less aggressive on the send side
  - Every iteration of pre copy

Stream Consumers

- Precopy stream
- Page In stream
- Disk copy stream

Stream Find Work

- Check each channel for work
  - VMotionStreamChannel_PollIn - READ SIDE - Handler
  - VMotionStreamChannel_PollOut - WRITE SIDE - Generator
- Check stream consumers for work
  - Check channels for completion work
    - VMOTION_STREAM_WORK_WRITE_COMPLETE
    - VMOTION_STREAM_WORK_READ_COMPLETE
  - Check streams for generate and Handle
    - VMOTION_STREAM_WORK_HANDLE
    - VMOTION_STREAM_WORK_GENERATE - dataCount > 0
- No work -> Wait -> VMotionStreamWait -> World_Wait
- Completions have priority over handle/generate

Lists the data structures used by STREAM Core

STREAM - DATA STRUCTURES

Data Structures

- VMotionMsgPreCopyData
  - Array of pgNums - VMOTION_MAX_PRE_COPY_SIZE - 1024
- VMotionPageGroupPublic
  - Array of pgNums/ pageInfo - VMOTION_MAX_PRE_COPY_SIZE_PER_MSG - 128
- VMotionRefCountedMemBlock
  - Holds all the page content in one big buffer – Write completion data!
- VMotionBuf
  - List of message buffer containing pointers to data in mem block

STREAM Data Structures

- VMotion Stream Channel
  - Channel Type -> Socket
  - Channel Type -> Disk
- VMotion Stream Channel Socket
  - Transmit - Pointer to VMotionVBuf
  - Recv - Pointer to IOVec -> RCD
- VMotion Channel Completion Data
  - Pointer to read/write completion data
- VMotion Read Completion Va
  - Pointer to one page's data – MPN/ VA etc
- VMotion Read Completion Data - RCD
  - Array of read of completion va's -VMOTION_READ_COMPLETION_MAX_VAS = 16 (64kb)

Walk through the precopy stream’s generate and handle routines

PRE COPY STREAM – AN EXAMPLE

Generator

- Stream Generation
  - Heavy lifting
  - Generator Function - VMotionSend_GetPrecopyPages
  - Populate VMotionPageGroupPublic from VMotionMsgPreCopyData
  - Compute to total size from PageGroupPublic and allocate ChannelBuf (MemBlock)
  - MemBlock comes from Stream Memory (16 MB)
  - Copy all page content to MemBlock
  - All data to be transmitted is prepared

Write Completion

- Write Completion
  - Lite weight
  - Write Completion function - VMotionSend_WriteCompletePages
  - Allocate external message buffers (socket mem) and make it to point to MemBlock
  - Link all the message buffers in VMotionVBuf and assign to the Socket.
  - Transmit the VMotionVbuf in the socket
  - Wait for transmit complete call back to free MemBlock

Generator Data Structures

- Generator
  - Heavy
  - VMotionMsgPreCopyData (VPC) -> VMotionPageGroupPublic
  - VMotionPageGroupPublic -> channelBuffer
  - channelBuffer -> VMotionMemBlock

- Write Completion
  - Lite
  - VMotionRefCountedMemBlock -> Stream Channel Socket
    -> VMotionVBuf
    -> MBuf_List

Handler

- VMotionStreamHandle
  - Lite weight
  - Read Header from the wire -> VMotionPageGroupPublic
  - Handler function = VMotionRecv_HandlePages
  - Process precopy header
  - Go over to VMotionPageGroupPublic num pages
  - Calculate channelBytes = num pages * PAGE_SIZE
  - Setup the channel completion data for reading from the wire

STREAM Complete Read

- VMotionStreamCompleteRead
  - Heavy Lifting

- 3 steps
  - Prepare (allocate) Pages for reading data from wire
  - Read from Wire
    - Copy page data out from the socket buffer to allocated pages
  - Add pages to the dest VM
    - Copy the page content read from the wire to the VM

1 – Read Preparation

- Read preparation function
  - VMotionRecv_ReadPreparePages
  - Takes a VMotionPageGroupPublic and prepares read completion data.
  - Loop over PageGroupPublic -> numpages
  - Prepare VA for each pgNum
    - Wait for low memory state to clear
    - VMotionRecvWaitForLowMem
    - Allocate a MPN
    - VmMemMigrate_GetAddPageMPN
    - Map the MPN with a Virtual Address -> Direct Mapping
  - Add the VA to read completion data
  - Read completion data is nothing but array of Vas
    - It has the total no of bytes to read from the wire

2 – Read from Wire

- VMotionStreamChannel_ReadCompletion
  - The read completion data has the VAs setup from reading
  - Setup the channel socket with these VAs
  - Net_SetIovec(channel->typeData.sock.iovec = va)
  - The read the pages off the wire in to the Vas
  - VMotionUtilWaitForUio
    - Loop around till channel bytes are read
    - MigrateNet_SetCallbackTarget - Setup low water bytes to be available to read
    - MigrateNet_WaitForSocketCallback - Wait for low water bytes to be available to read
    - Actual Read - MigrateNet_ReceiveUio

3- Read Completion

- VMotionRecv_ReadCompletePages
  - Heavy Weight
  - Loop over VMotionPageGroupPublic -> numPages
  - Get the corresponding VA which has the page content
  - Wait for low memory – VMotionRecvWaitForLowMem
  - VmMemMigrate_AddPage
    - Add the page to destination VM using pgNum and pageInfo->type
    - MIGRATE_PAGE_UNMAPPED
    - MIGRATE_PAGE_COMPRESSED
    - MIGRATE_PAGE_ZERO
    - MIGRATE_PAGE_SWAPPED
    - MIGRATE_PAGE_REGULAR - VmMem_AllocAndCopyPage
    - MIGRATE_PAGE_COW

Handler – Data Structures

- Handler
  - Lite
  - Read from wire -> VMotionPageGroupPublic
  - VMotionPageGroupPublic -> Channel Completion Data

- Read Completion
  - Heavy
  - Channel Completion Data -> VMotionReadCompletionData
  - VMotionReadCompletionData -> Streamchannel->typeData.sock.iovec
  - Read from Wire -> Streamchannel->typeData.sock.iovec
  - VMotionReadCompletionData -> Commit to Dest VM memory