Address Range Memory Mirroring

July 13, 2016
Taku Izumi
FUJITSU LIMITED
Table of Contents

- Address Range Mirroring overview
  - What is Address Range mirroring

- Current status of linux
  - Current implementation for Address Range mirroring

- Future plan
  - Feedback from MM summit 2016
Address Range Mirroring overview
Memory Mirroring

- Memory RAS feature on Xeon family-based systems
- Provides memory redundancy
  - Data writing
    - written to both sides of the memory mirror at the same time (Memory A & B)
  - Data reading
    - Memory Controller reconfirms data validity by comparing data
    - If an uncorrectable error is detected in Memory A, data in Memory B is used for the read operation [tolerance for UCE]
Traditional Memory Mirroring

- **Full Mirror Mode**
  - Pros: Transparent to OS
  - Cons: Halves memory capacity available to OS

- **Partial Mirror Mode**
  - Pros: Transparent to OS and Keep More capacity than Full Mirror Mode
  - Cons: Mirrored range is one-sided

![Diagram showing Full Mirror Mode and Partial Mirror Mode]
Address Range Mirroring

- New memory RAS feature on Haswell EX based systems
- Allows high granularity of mirroring
  - Configurable the amount of mirrored memory size
    - optimize total available memory while keeping highly reliable memory range
- Distributes mirrored memory range on each NUMA node
  - Keeps NUMA performance

OS View of physical memory

Address Range Mirror

- capacity in case of full mirror
- Actual memory capacity
Address Range Mirroring cont.

- Requires OS support to fully utilize Address Range Mirroring
  - Necessary to be aware of mirrored region

- Provides Firmware-OS interface
  - UEFI Variables
    - A method to request the amount of mirrored memory
  - UEFI Memory map
    - Presents mirrored memory range on the platform
Mirrored memory size configuration

- UEFI variable is used to configure mirroring
  - MirrorRequest
    - Written by the OS to request a new mirror configuration on the next boot
  - MirrorCurrent
    - Written by the firmware and read by the OS to communicate the current status of Address Range Mirroring

```c
typedef struct {
  UINT8     MirrorVersion;
  BOOLEAN   MirrorMemoryBelow4GB;  // set to true to mirror memory below 4 GB
  UINT16    MirroredAmountAbove4GB;  // percentage of memory to mirror above 4GB
  UINT8     MirrorStatus;
} ADDRESS_RANGE_MIRROR_VARIABLE_DATA
```
latest efibootmgr(8) supports UEFI variables for Address Range Mirroring

- `m`: set ‘t’ to mirror memory below 4GB
- `M`: percentage memory to mirror above 4GB

```
# efibootmgr -m t -M 25.00
```

Confirm current settings of mirroring

```
# efibootmgr
BootCurrent: 0002
Timeout: 10 seconds
BootOrder: 0002, 0001, 0000
Boot0000* EFI SCSI Device
Boot0001* EFI Internal Shell
Boot0002* opensuse-secureboot
MirroredPercentageAbove4G: 25.00
MirrorMemoryBelow4GB: true
```
Presentation method of mirrored range

The information which address range is mirrored is passed via EFI memory map

<table>
<thead>
<tr>
<th>Type</th>
<th>Start</th>
<th>End</th>
<th># Pages</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>BS_data</td>
<td>0000000000000000</td>
<td>-0000000000000FFF</td>
<td>0000000000000001</td>
<td>00000000000001000F</td>
</tr>
<tr>
<td>available</td>
<td>0000000000010000</td>
<td>-000000000003FFFF</td>
<td>000000000000003F</td>
<td>00000000000001000F</td>
</tr>
<tr>
<td>BS_data</td>
<td>0000000000040000</td>
<td>-000000000009FFFF</td>
<td>0000000000000060</td>
<td>00000000000001000F</td>
</tr>
<tr>
<td>BS_data</td>
<td>0000000000010000</td>
<td>-000000000006FFFF</td>
<td>00000000000000F0</td>
<td>00000000000001000F</td>
</tr>
<tr>
<td>available</td>
<td>0000000000100000</td>
<td>-000000000035FC5F</td>
<td>000000000000034F</td>
<td>00000000000001000F</td>
</tr>
<tr>
<td>RT_data</td>
<td>000000000035FC6000</td>
<td>-000000000035FCCFFF</td>
<td>0000000000000007</td>
<td>80000000000001000F</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BS_data</td>
<td>000000000047ED2000</td>
<td>-00000000048013FF</td>
<td>0000000000000142</td>
<td>00000000000001000F</td>
</tr>
<tr>
<td>available</td>
<td>0000000100000000</td>
<td>-0000000203FFFFFF</td>
<td>0000000001F40000</td>
<td>00000000000001000F</td>
</tr>
<tr>
<td>available</td>
<td>0000000204000000</td>
<td>-0000000302FFFFFF</td>
<td>0000000001FF0000</td>
<td>00000000000001000F</td>
</tr>
<tr>
<td>available</td>
<td>0000000303000000</td>
<td>-0000000304FFFFFF</td>
<td>0000000002000000</td>
<td>00000000000001000F</td>
</tr>
<tr>
<td>reserved</td>
<td>0000000005FC0000</td>
<td>-0000000005FFFFFF</td>
<td>0000000000040000</td>
<td>00000000000001000F</td>
</tr>
<tr>
<td>reserved</td>
<td>0000000006000000</td>
<td>-0000000008FFFFFF</td>
<td>0000000000030000</td>
<td>80000000000001000F</td>
</tr>
</tbody>
</table>

- **EFI_MEMORY_MORE_RELIABLE** attribute: 0x10000

- **EFI_MEMORY_MORE_RELIABLE** attribute in EFI Memory descriptor indicates mirrored range
  - Defined in UEFI spec 2.5
Motivation

Background

- Linux can process memory errors in user space memory
  - Just kill the affected processes, even recover if the broken page can be replaced by reading from disk
  - Avoid broken page in the future
- Linux has no recovery path for errors encountered during kernel code execution
  - Uncorrectable Error in kernel memory would crash the system
- Full memory mirroring is a good approach, however, as system memory capacity grows, the amount of memory lost for redundancy also grows

Motivation

- Improve high availability by avoiding uncorrectable errors in kernel memory
  - Allocate all kernel memory from mirrored memory
Current Status of Linux
Physical memory management

- **Memblock**
  - Manage memory blocks during early bootstrap period
  - Discarded after initialization and take over to Zone allocator

- **Zone Allocator**
  - Usual kernel memory allocator
Memblock

- Simply manages memory blocks
  - Consists of two arrays
    - memory: All the present memory in the system
    - reserved: Allocated memory ranges
  - Allocate by finding regions in memory && !reserved
Mirror support of Memblock has been merged into linux-4.3

- Find mirrored region from EFI memory map information
  - Mark as MEMBLOCK_MIRROR
- Try to allocate from mirrored region
  - If run out of mirrored memory, fall back to use non-mirrored memory

Scan the memory map and mark any mirrored regions

Try to allocate from mirrored region
Zone allocator

- Manages memory areas called zones
  - All pages are managed by Zone
    - ZONE_DMA, ZONE_DMA32
      - used for DMA
    - ZONE_NORMAL
      - memory directly mapped, used by kernel and user space
- Find zones suitable for memory allocation and allocate memory

- As of linux-4.3, no mirror support for zone allocator
  - Works without any regard to mirrored region
Solution for mirror support of Zone allocator

■ Requirement
  ■ Allocate kernel memory from mirrored region
  ■ Allocate user memory from non-mirrored region

■ Clue
  ■ ZONE_MOVABLE
    • Not exist by default
      • need to boot with “kernelcore” or “movablecore” specified
    • migratable memory can be allocated = users page only
      • kernel pages won’t be allocated; kernel page is NOT migratable
    • go well together memory hot-remove

■ Idea
  ■ Arrange non-mirrored range into ZONE_MOVABLE
kernelcore=mirror boot option

- New in linux-4.6 for Address Range Mirroring
- By specifying “kernelcore = mirror” boot option,
  - Non-mirrored region will be arranged into ZONE_MOVABLE
  - Kernel memory won’t be allocated from ZONE_MOVABLE, so it will be allocated from mirrored region

![Diagram showing memory zones and mirrored regions](image_url)
Drawback of kernelcore=mirror approach

■ When running out of mirrored memory, never fall back to use non-mirrored memory
  ■ Possible that kernel memory is exhausted, though there is non-mirrored memory available
  ■ This behavior is as planned to protect kernel memory surely

■ Need sizing of total kernel memory

![Diagram showing memory allocation and kernel data demand]

- **NORMAL**: 
  - No room for demand size
- **MOVABLE**: 
  - Kernel memory won’t be allocate from ZONE_MOVABLE

Copyright 2016 FUJITSU LIMITED
Future Plan
Future plan

■ Handling of user’s memory
  ■ Prevent involuntarily user’s memory allocation from mirrored range
    • Currently user’s memory can be allocated from ZONE_NORMAL
    • Add a new __GFP_NONMIRROR allocation flag to be part of GFP_HIGHUSER_MOVABLE ?

  ■ Add the method that any user apps’ memory can be allocated from mirrored memory
    • Add a new MADV_MIRROR flag to the madvice(2) ?

■ Handling of mirrored memory exhaustion case
  ■ Add fallback to non-mirrored memory option
    • In my opinion, we should not fallback. Change mirrored size to expand instead
Feedback from MM summit 2016

- Got negative feedback for putting user-space program into mirrored range
  - madvice(2) is wrong interface
    - placement in mirrored memory would be mandatory
    - mirrored memory could be an opt-out resource rather than opt-in
      - But nobody would volunteer to opt-out…
  - A little messy so everything has to go there including shared libraries
    - Need restart apps?
  - Sizing for ZONE_NORMAL becomes difficult
    - second coming of low-memory problem
  - Assuming user-space program can figure the right thing to choose what needs to be mirrored is not safe
    - Security issues: some program can force the exhaustion of mirrored memory

- Partial mirroring is simply the wrong approach
  - Simply mirroring the entire address space is easy
Summary

- Address Range Mirroring overview
  - What is Address Range mirroring

- Current status of linux
  - Current implementation for Address Range mirroring

- Future plan
  - Feedback from MM summit 2016
Fujitsu booth is at corner of the foyer. We are looking forward to see you in our booth.

Fujitsu shares the emerging technology and trends. Please come and experience the future innovation of technology with us. Find out what Fujitsu delivers you today.

Cloud Monitoring software for OpenStack (based on Monasca)
- Fujitsu Software ServerView Cloud Monitoring Manager

Cloud Service Management software (Open Source Software)
- Fujitsu Software Enterprise Service Catalog Manager