Introduction to ACPI Based Memory Hot-Plug

Tang Chen
<tangchen@cn.fujitsu.com>
Agenda

1. Why need Memory Hot-Plug
2. ACPI & Memory Hot-Plug
3. Memory hot-add
4. Memory hot-remove
5. Movable node
6. Bootmem handling
7. Future work
Why need memory Hot-Plug

1. Balance the load.
2. Reduce power consumption.
3. Handle hardware error.
Why need memory Hot-Plug

4. Guest OS should support memory hotplug.
   - VMware supports virtual memory device hotplug for the virtual machine.
   - The similar feature is being developed for KVM.

5. ACPI provides sufficient conditions
   - With the help of ACPI, hardware and firmware are now able to support memory hotplug physically.
Agenda

1. Why need Memory Hot-Plug
2. ACPI & Memory Hot-Plug
3. Memory hot-add
4. Memory hot-remove
5. Movable node
6. Bootmem handling
7. Future work
ACPI & Memory Hot-Plug

• ACPI: Advanced Configuration and Power Interface

ACPI is an interface specification of Operating System-directed motherboard device configuration and Power Management.

--- ACPI Specification 5.0

OS layer framework.
✓ Event handling
✓ API

Dynamic methods used at run time.
✓ _EJ0
✓ _STA
✓ ......

SCI (System Control Interrupt)

Kernel (Software)

ACPI Driver

Methods (dynamic)

ACPI Tables (static)

ACPI BIOS (Firmware)

Run Time

Boot Time

ACPI Registers

Hardware

Static info used only at boot time.
✓ DSDT
✓ SRAT
✓ ......

Event driven model.
✓ Event registers
✓ Control registers
✓ ......
ACPI & Memory Hot-Plug

• ACPI and Memory Hot-Plug

Kernel

Memory Device Driver

Memory Hot-Plug Subsystem

ACPI Driver

Methods (dynamic)

ACPI Tables (static)

ACPI BIOS

Hardware

ACPI Registers

Call ACPI Method

Read ACPI Tables

Call API

Install event handler

Call event handler

Call device dependent code

Call ACPI Method

Hardware operation

Hot-Plug happens

Event info

Boot time process

Run time process

Generate SCI
(System Control Interrupt)
ACPI & Memory Hot-Plug

• Main jobs of Memory Hot-Plug

- Node data
- Direct mapping
- Virtual memory mapping
- Page online & offline
- Page migration
- Event handler
ACPI & Memory Hot-Plug

- Things associated with Memory Hot-Plug

1. Memory block to be hot-plugged.
2. User processes’ pagetable.
4. Virtual memory mapping pages.
5. Virtual memory mapping pagetable.
Agenda

1. Why need Memory Hot-Plug
2. ACPI & Memory Hot-Plug
3. Memory hot-add
4. Memory hot-remove
5. Movable node
6. Bootmem handling
7. Future work
Memory hot-add

- Generic memory definition & sparse memory model

**Generic memory definition** divides a memory range into several blocks. (128MB per block by default on x64)

**Sparse memory model** divides a memory range into several sections, in which the memory is contiguous. (128MB per section, one section per block by default on x64)
Memory hot-add

- Add memory (1)

Blocks in the memory range are hot-added one by one.
Memory hot-add

- Add memory (2)

1. Initialize direct mapping pagetable.

2. Allocate virtual memory mapping pages.

3. Initialize virtual memory mapping pagetable.
Memory hot-add

• Add memory (3)

The newly added pages are offline and not present.

```
echo online > /sys/devices/system/memory/memoryX/state
```
Memory hot-add

- Online pages
Memory hot-add

• Configuration
  – mm/Kconfig

```c
config MEMORY_HOTPLUG
  bool "Allow for memory hot-add"
  depends on SPARSEMEM || X86_64_ACPI_NUMA
  depends on HOTPLUG && ARCH_ENABLE_MEMORY_HOTPLUG
  depends on (IA64 || X86 || PPC_BOOK3S_64 || SUPERH || S390)
```
Agenda

1. Why need Memory Hot-Plug
2. ACPI & Memory Hot-Plug
3. Memory hot-add
4. Memory hot-remove
5. Movable node
6. Bootmem handling
7. Future work
Memory hot-remove

- Remove memory (1)
Memory hot-remove

• Remove memory (2)

1. Unmap user pages.
   • Kernel will generate a page fault for each process who access these pages, and the process will wait till the migration is over.


3. Copy data from old pages to new pages.
Memory hot-remove

- Remove memory (3)

4. Update user processes’ pagetable.
   - Also wake up all the processes waiting for these pages.

5. Isolate old pages.
   - Pages are not in the buddy system, and won’t be allocated to anyone.

6. Set the block state to offline.

4. Update user processes’ pagetable.
   - Also wake up all the processes waiting for these pages.

5. Isolate old pages.
   - Pages are not in the buddy system, and won’t be allocated to anyone.

6. Set the block state to offline.
Memory hot-remove

• Remove memory (4)

6. Free kernel direct mapping pagetable.

7. Free virtual memory mapping pages.

8. Free virtual memory mapping pagetable.
Memory hot-remove

- Remove memory (5)

Kernel space

- module mapping space
- kernel text mapping
- virtual memory map (1TB)
- direct mapping (64TB)
- vmalloc/ioremap space

Physical space

- page_structs
- pages (used)
- removed
- pages (used)

User space

- process 1 (128TB)
- process 2 (128TB)
- process 3 (128TB)
- process i (128TB)
- process n (128TB)
Memory hot-remove

• Post work: automatically remove the node

Node hot-remove

- Set node state to offline
- Remove /sys files
- Free wait_table
- Clear pgdat
Memory hot-remove

• Merged into Linux 3.9
• Configuration
  – mm/Kconfig

```c
config MEMORY_HOTREMOVE
  bool "Allow for memory hot remove"
  select MEMORY_ISOLATION
  select HAVE_BOOTMEM_INFO_NODE if X86_64
  depends on MEMORY_HOTPLUG && ARCH_ENABLE_MEMORY_HOTREMOVE
  depends on MIGRATION
```
Memory hot-remove

- Kernel pages cannot be hot-removed
Agenda

1. Why need Memory Hot-Plug
2. ACPI & Memory Hot-Plug
3. Memory hot-add
4. Memory hot-remove
5. Movable node
6. Bootmem handling
7. Future work
Movable node

- **NUMA: Non-Uniform Memory Access**
  - A node consists of a set of CPUs and memory.
  - CPUs access memory in the same node faster.
  - Meaningful in 64bits platform only. 32bits platform has only one node.
Movable node

- Zone: Different types of memory in a node

- **ZONE_DMA / ZONE_DMA_32 (64bits only):** used for DMA.

- **ZONE_NORMAL:** Memory directly mapped, used by kernel and user space.

- **ZONE_HIGHMEM:** For 32bits kernel to access memory not directly mapped.

- **ZONE_MOVABLE:** Memory can be migrated. (user space only)
Movable node

• Problem

Kernel allocates ZONE_NORMAL on each node evenly

Kernel may use ZONE_NORMAL

Each node has ZONE_NORMAL

No node is hot-removable

un-hot-removable

ZONE_DMA
ZONE_DMA32
ZONE_NORMAL
ZONE_MOVABLE

ZONE_NORMAL
ZONE_MOVABLE

ZONE_NORMAL
ZONE_MOVABLE

ZONE_NORMAL
ZONE_MOVABLE

ZONE_NORMAL
ZONE_MOVABLE
Movable node

• Solution

Configure a node to have only ZONE_MOVABLE.

Kernel can not use ZONE_MOVABLE.

Movable node has no ZONE_NORMAL.

The node is hot-removable.
Movable node

• Static configuration
  – SRAT: System Resource Affinity Table

Static information of NUMA architecture.
Movable node

• movablecore = **acpi** (New)

- Use SRAT to arrange ZONE_MOVABLE.
- Only for memory hotplug users.
- Still being pushing.

**SRAT memory affinities**

- Node 0: unhotpluggable
  - node 0:
    - ZONE_DMA
    - ZONE_DMA32
    - ZONE_NORMAL
- Node 1: hotpluggable
  - node 1:
    - ZONE_MOVABLE
- Node i: hotpluggable
  - node i:
    - ZONE_MOVABLE
- Node n: unhotpluggable
  - node n:
    - ZONE_NORMAL
Movable node

- The old way (no performance lost)
  - kernelcore / movablecore = nn \{G|M|K\} (Old)

  - Allocate ZONE_MORMAL in each node evenly.
  - For regular users.
**Movable node**

- **Dynamic configuration**

  offline and online again

1. **online_kernel (NEW)**
   - Set to ZONE_NORMAL.

2. **online_movable (NEW)**
   - Set to ZONE_MOVABLE.

3. **online (Improved)**
   - Keep the previous state.
   - (ZONE_NORMAL for the first time)

**Rule:** ZONE_MOVABLE should always be after ZONE_NORMAL, never overlaps.
Movable node

• Drawback

No good enough way to solve this problem now.
Movable node

• Merged into Linux 3.8
• Configuration
  – mm/Kconfig

```c
config MOVABLE_NODE
  boolean "Enable to assign a node which has only movable memory"
  depends on HAVE_MEMBLOCK
  depends on NO_BOOTMEM
  depends on X86_64
  depends on NUMA
  default n
```
Agenda

1. Why need Memory Hot-Plug
2. ACPI & Memory Hot-Plug
3. Memory hot-add
4. Memory hot-remove
5. Movable node
6. Bootmem handling
7. Future work
Bootmem handling

• Memblock: A bootmem allocator
  – Consists of two arrays

memblock.memory[]: All the present memory in the system.

memblock.reserve[]: Allocated memory.

Free unused memory to buddy system at last.
Bootmem handling

• Problem at boot time
  – Bootmem allocator memblock may allocate hotpluggable memory for kernel at boot time.

1. Memblock is ready.
2. Parse SRAT (too late).
3. Initialize ZONEs.

Hotpluggable memory ranges are unknown.

Allocated by memblock

Not hot-removable
Bootmem handling

• Solution

1. Parse SRAT earlier, before memblock starts to work.
2. Introduce flags into memblock, and reserve hotpluggable memory with a special flag in memblock.
Bootmem handling

1. Memblock is ready.

```
memblock.memory[]
```

```
node0
  low memory
  memory1

node1
  memory2
  memory3

node2
  memory4
  memory5

node3
  memory6
  memory7
```

All the memory ranges in the system are put into memory[].

Allocated memory ranges will be put into reserve[].

```
memblock.reserve[]
```

Boot time
Bootmem handling

2. Before parsing SRAT.

- Reserve kernel _data, _text, setup data, …, with flag DEFAULT.
- Any node the kernel resides in is unhotpluggable.
  (Not necessary to be node 0)
- No new memory allocation, so no hotpluggable memory could be used by the kernel.
Bootmem handling

3. Parsing **SRAT**.

**Reserve** *hotpluggable* memory with flag **HOTPLUGGABLE**.

**memblock.reserve[]**
Bootmem handling

4. After parsing SRAT, **hotpluggable** memory has been reserved.

- **memblock.memory[]**

- **node0**
  - low memory
  - memory1

- **node1**
  - memory2
  - memory3

- **node2**
  - memory4
  - memory5

- **node3**
  - memory6
  - memory7

- **No** hotpluggable memory used by kernel.

- **hottpluggable memory**
  - DEFAULT
  - membblock.reserve[]

- **unhotpluggable memory**
  - DEFAULT
Bootmem handling

5. Memory initialization has been finished.

Free hotpluggable memory to buddy system. (NEW)

DEFAULT memblock.reserve[]
Agenda

1. Why need Memory Hot-Plug
2. ACPI & Memory Hot-Plug
3. Memory hot-add
4. Memory hot-remove
5. Movable node
6. Bootmem handling
7. Future work
Future work

• Node local pagetable and vmemmap.
  – Improve performance.

• Migrate user pages pinned in memory.
  – For those who pin pages for a long time.

• User space tools, like libnuma and numactl.
  – A library of functions.
  – Commands.
Thank you!

Q&A
Movable node

- Performance tests

Alloc: 40% down
Read: 33% down
Write: 12% down

![Time of accessing 20GB memory (s)](chart)

- Alloc: 40% down
- Read: 33% down
- Write: 12% down