Linux & Memory Maps

Presented by: Abdul Qadeer

Presented in Open Source Series Workshop 2010

22-24 December, 2010 © ICOSST 2010
Don’t confuse physical & virtual address spaces

• Virtual address space
  – No one understands that outside the processor!

• Physical address space
  – You need that to do memory operations and many other I/O operations
int x = 25;
int address_of_x = &x;

The C / C++ address of operator (&). Does it return a virtual address or a physical address?
Don’t confuse physical & virtual address spaces

• VAS > PAS
  – Probably a normal scenario

• VAS == PAS
  – E.g. 32 bit VAS and 4GiB installed DRAM
  – Some DRAM might go un-used and hence wasted!

• VAS < PAS
  – Not a common scenario
  – Might be used to physically isolate DRAM used by programs
  – Physical address extension; 32 bit VAS, 36bit PAS
Don’t confuse physical & virtual address spaces

Can look VAS and PAS in /proc/cpuinfo

```bash
root@HPCNL-SR3-64bit:~# dmidecode -s sys insanlarınıs
CPU MHz: 1000.000
Cache size: 1024 KB
Physical id: 1
Siblings: 2
Core id: 1
Cpu cores: 2
Apic id: 3
Initial apic id: 3
Fpu: yes
Fpu exception: yes
Cpuid level: 1
WP: yes
Flags: fpu vme de pse tsc msr pae mce cx8 apic mtrr pge mca

```

Legacy svm extapic cr8 Legacy
Bogomips: 1995.11
TLB size: 1024 4K pages
Clflush size: 64
Cache alignment: 64
Address sizes: 40 bits physical, 48 bits virtual
Power management: ts fid vid ttp tm stc
```
Memory mapped I/O

- Part of PAS assigned to a device
  - Writing to this mem => writing to device
  - Reading from this mem => reading from device
  - Devices negotiate for their required MMI/O region
    - Old, dumb devices (e.g. ISA devices)
    - Newer, smarter devices (e.g. PCIe devices)
    - DMA concerns

- MMI/O usage needs special care
  - No compiler optimization (e.g. by using –O0)
  - No pipeline reordering (use Linux read / write barriers)
  - Disable caching (e.g. using bits in TLB)
Memory mapped I/O

• MMI/O PAS not usable as normal DRAM
  – MMI/O does not take away actual DRAM!
  – But does take away a region of PAS
  – The corresponding DRAM might go wasted

• Relocate actual DRAM to elsewhere
  – VAS should be big enough for relocation
  – Chipset should allow relocation
Memory mapped I/O

Can look PAS allocation in /proc/iomem

```
root@HPCNL-SR4-64bit~:

0000:00:04.0
ee000000-ee000000: PCI Bus 0000:0a
  ee0fe000-ee0ffffff : 0000:0a:0e.0
  ee0fe000-ee0ffffff : sata_swv
0000:00:00.0
ee300000-ee3fffff : PCI Bus 0000:0e
  ee39c000-ee39ffff : 0000:0e:00.0
  ee39c000-ee39ffff : ixgbe
  ee3a0000-ee3bffff : 0000:0e:00.0
  ee3a0000-ee3bffff : ixgbe
  ee3c0000-ee3fffff : 0000:0e:00.0
  ee3c0000-ee3fffff : ixgbe
f0000000-f0ffffff : reserved
  f0000000-f3ffffff : PCI MMCONFIG 0000 [bus 00-3f
  f0000000-f1ffffff : pnp 00:0a
  f4000000-f7ffffff : GART
  fe000000-ffffff : reserved
    fec0000-fec0003ff : IOAPIC 0
    fec01000-fec013ff : IOAPIC 1
    fec02000-fec023ff : IOAPIC 2
    fec00000-fec003ff : HPET 0
    fec00000-fec000ff : Local APIC
10000000-22ffffff : System RAM
[root@HPCNL-SR4-64bit ~]#
```
Processor, North & South Bridges

Picture taken from Gustavo Duarte blog accessible at: http://duartes.org/gustavo/blog/post/motherboard-chipsets-memory-map
Picture taken from Gustavo Duarte blog accessible at: http://duartes.org/gustavo/blog/post/motherboard-chipsets-memory-map
Picture taken from Gustavo Duarte blog accessible at: http://duartes.org/gustavo/blog/post/motherboard-chipsets-memory-map
Presented in Open Source Series Workshop 2010

Picture taken from Gustavo Duarte blog accessible at: http://duartes.org/gustavo/blog/post/motherboard-chipsets-memory-map
MTRR registers

• Memory region registration process
  – MTTR registers used
  – BIOS tells some regions initially
  – OS tries to find out the optimal registration

• Looking Linux log in dmesg
  – Shows interesting system information
MTRR registers

Initialization of the MTRR registers:

Linux version 2.6.33.3-85.fc13.x86_64 (mockbuild@x86-02.phx2.fedoraproject.org) (gcc version 4.4.4 20100503 (Red Hat 4.4.4-2) (GCC) ) #1 SMP Thu May 6 18:09:49 UTC 2010
Command line: ro root=UUID=84b584b8-db6c-40c8-88ce-9e618eb27b7f4 rd_NO_LUKS rd_NO_LVM rd_NO_MD rd_NO_DM LANG=en_US.UTF-8 SYSFONT=latarcyrheb-sun16 KEYTABLE=us rhgb quiet

BIOS-provided physical RAM map:
- BIOS-e820: 0000000000000000 - 0000000000000000 (usable)
- BIOS-e820: 0000000000000000 - 0000000000000000 (usable)
- BIOS-e820: 0000000000000000 - 0000000000000000 (reserved)
- BIOS-e820: 0000000000000000 - 0000000000000000 (ACPI data)
- BIOS-e820: 0000000000000000 - 0000000000000000 (reserved)
- BIOS-e820: 0000000000000000 - 0000000000000000 (reserved)
- BIOS-e820: 0000000000000000 - 0000000000000000 (reserved)
- BIOS-e820: 0000000000000000 - 0000000000000000 (usable)

NX (Execute Disable) protection: active
DMI 2.4 present.
No AGP bridge found
last_pfn = 0x230000 max_arch_pfn = 0x40000000
MTRR default type: uncacheable
MTRR fixed ranges enabled:
- 00000-9FFFF write-back
- A0000-BFFFF uncachable
- C0000-D3FFF write-protect
- D4000-EBFFF uncacheable
- EC000-FFFF write-protect
MTRR variable ranges enabled:
- 0 base 0000000000000000 mask FF80000000 write-back
- 1 base 0000000000000000 mask FFC0000000 write-back
- 2 base 0000000000000000 mask FFF0000000 write-back
- 3 base 0100000000000000 mask FF00000000 write-back
MTRR registers

```
root@HPCLP-64bit-:~ # cat /sys/devices/system/node/node0/pmem

SRAT: Node 0 PXM 0 0-a0000
SRAT: Node 0 PXM 0 100000-d0000000
SRAT: Node 0 PXM 0 10000000-130000000
SRAT: Node 1 PXM 1 130000000-230000000
NUMA: Allocated memnode map from 1100 - 15640
NUMA: Using 20 for the hash shift.
Bootmem setup node 0 0000000000000000-0000000130000000
    NODE_DATA [000000000015640-0000000002b63f]
    bootmap [000000000002c000-0000000000051fff] pages 26
    (14 early reservations) ==> bootmem [0000000000-0130000000]
    #0 [0000000000-000001000]  BIOS data page  ==> [0000000000-000001000]
    #1 [0001000000-0001db92c8]  TEXT DATA BSS ==> [0001000000-0001db92c8]
    #2 [003743a000-0037ef3c6]  RAMDISK ==> [003743a000-0037ef3c6]
    #3 [0001dba000-0001dba2d9]  BRK ==> [0001dba000-0001dba2d9]
    #4 [000000e720-0000010000]  BIOS reserved ==> [000000e720-0000010000]
    #5 [000000e710-000000e720]  MP-table mpf ==> [000000e710-000000e720]
    #6 [000000e700-000000e700]  BIOS reserved ==> [000000e700-000000e700]
    #7 [000000f344-000000f710]  BIOS reserved ==> [000000f344-000000f710]
    #8 [000000f000-000000f034]  MP-table mpc ==> [000000f000-000000f034]
    #9 [0000001000-0000003000]  TRAMPOLINE ==> [0000001000-0000003000]
    #10 [0000003000-0000007000]  ACPI WAKEUP ==> [0000003000-0000007000]
    #11 [0000008000-000000c000]  PGTABLE ==> [0000008000-000000c000]
    #12 [000000c000-0000010000]  PGTABLE ==> [000000c000-0000010000]
    #13 [0000010000-0000015640]  MEMNODEMAP ==> [0000010000-0000015640]
Bootmem setup node 1 0000000130000000-0000000230000000
    NODE_DATA [0000000130000000-0000000130015fff]
    bootmap [0000000130016000-0000000130035fff] pages 20
    (14 early reservations) ==> bootmem [0130000000-0230000000]
    #0 [0000000000-000001000]  BIOS data page
    #1 [0001000000-0001db92c8]  TEXT DATA BSS
    #2 [003743a000-0037ef3c6]  RAMDISK
```
MTRR registers

```
root@HPNL-SM-64bit:~#

sizeof(inode) = 584 bytes
sizeof(dentry) = 192 bytes
sizeof(ext3inode) = 792 bytes
sizeof(buffer_head) = 104 bytes
sizeof(skbuff) = 232 bytes
sizeof(task_struct) = 5968 bytes
devtmpfs: initialized
regulator: core version 0.5
Time: 9:42:22 Date: 12/08/10
NET: Registered protocol family 16
node 0 link 1: io port [c000, efff]
TOM: 00000000d0000000 aka 3328M
node 0 link 1: mmio [d8000000, e00fffff]
node 0 link 1: mmio [e8000000, ee3fffff]
node 0 link 1: mmio [f0000000, ff000000]
node 0 link 1: mmio [a0000, bfffff]
TOM2: 0000000230000000 aka 8960M
bus: [00, fe] on node 0 link 1
bus: 00 index 0 io port: [0, ffff]
bus: 00 index 1 mmio: [d0000000, e7ffffff]
bus: 00 index 2 mmio: [e8000000, effffff]
bus: 00 index 3 mmio: [f0000000, fffffff]
bus: 00 index 4 mmio: [a0000, bfffff]
bus: 00 index 5 mmio: [230000000, fcf00000000000000]
ACPI FADT declares the system doesn't support PCIe ASPM, so disable it
ACPI: bus type pci registered
PCI: MMCONFIG for domain 0000 [bus 00-3f] at [mem 0xf0000000-0xf3ffffff] (base 0xf0000000)
PCI: MMCONFIG at [mem 0xf0000000-0xf3fffffff] reserved in E820
PCI: Using configuration type 1 for base access
bio: create slab <bio-0> at 0
ACPI: EC: Look up EC in DSDT
```
How an Intel processor starts from 0xFF FF FF F0

Reset vector

All grey stuff is memory mapped

Picture taken from Gustavo Duarte blog accessible at: http://duartes.org/gustavo/blog/post/how-computers-boot-up/
How an Intel processor starts from 0XFF FF FF F0

Reset vector
0xFFFFFFF0

JUMP to 0xF0000
(last 16 bytes of memory are addressable via EIP
thanks to power-up hack.)

Unaddressable memory, real mode is limited to 1 MB.
This region represents ~4 GB and is not to scale.

System BIOS
0xFFFF

Extended System BIOS
0xF0000

Expansion Area (maps ROMs for old peripheral cards)

Legacy Video Card Memory Access

Accessible RAM Memory (640KB is enough for anyone - old DOS area)

All grey stuff is memory mapped

Jump to main BIOS

This is due to X86 legacy

Picture taken from Gustavo Duarte blog accessible at: http://duartes.org/gustavo/blog/post/how-computers-boot-up/
VAS distribution among Kernel and a user space process

• 3G/1G split
  – Theoretically $2^{32}$ VAS available to a process
  – Kernel is mapped for every process for performance reasons
  – Can change this split at kernel compile time to say 2G/2G or by a patch to 4G/4G

• Half / Half split in 64 bit systems
  – 128 TB for processes and Kernel
VAS distribution among Kernel and a user space process

4GB
Kernel Mode (System)

3GB
User Mode (Application)

0

32bit System

64bit System

FF000000 FFFFFFFF
Canonical "higher half"
FF800000 00000000
Noncanonical addresses

00000000 FFFFFFFF
Canonical "lower half"
000007FF FFFFFFFF

Process address space

• Process can use almost any part of user-VAS
Process address space

- Process can use almost any part of user-VAS
- Application binary interface (ABI)
  - Conventions

Figure taken from Virtual Machines, versatile platforms for systems and processes, ISBN-10: 1558609105
Process address space

• Process can use almost any part of user-VAS
• Application binary interface (ABI)
  – Conventions
• Binary file format
  – ELF, COFF
• Language run-time
  – C, C++ etc.
• OS conventions
  – Loader, linker

Figure taken from Virtual Machines, versatile platforms for systems and processes, ISBN-10: 1558609105
Process address space

- Process can use almost any part of user-VAS
- Application binary interface (ABI)
  - Conventions
- Binary file format
  - ELF, COFF
- Language run-time
  - C, C++ etc.
- OS conventions
  - Loader, linker

Figure taken from Virtual Machines, versatile plattorms for systems and processes, ISBN-10: 1558609105
Guard pages

- Not everything in VAS is mapped
  - Address 0 is invalid
  - Guard page between stack and heap
  - Similar guard pages for thread stack
Guard pages

- See process virtual address map
  - `cat /proc/1/maps` or `cat /proc/self/maps`

<table>
<thead>
<tr>
<th>Address Range</th>
<th>Rights</th>
<th>Offset</th>
<th>Device</th>
<th>Inode File Associated</th>
<th>File Associated</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000-00400000</td>
<td>r-xp</td>
<td>00000000</td>
<td>08:03</td>
<td>5767213</td>
<td>/bin/cat</td>
</tr>
<tr>
<td>00600000-00600000</td>
<td>r-wp</td>
<td>00000000</td>
<td>08:03</td>
<td>5767213</td>
<td>/bin/cat</td>
</tr>
<tr>
<td>00600000-00600000</td>
<td>r-wp</td>
<td>00000000</td>
<td>00:00</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>00d00000-00d26000</td>
<td>r-wp</td>
<td>00000000</td>
<td>00:00</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>392c020000-392c21e000</td>
<td>r-xp</td>
<td>00000000</td>
<td>08:03</td>
<td>131764</td>
<td>[heap]</td>
</tr>
<tr>
<td>392f01e000-392f45f000</td>
<td>r-xp</td>
<td>00000000</td>
<td>08:03</td>
<td>131764</td>
<td>/lib64/ld-2.12.so</td>
</tr>
<tr>
<td>392f45f000-392f420000</td>
<td>r-wp</td>
<td>00000000</td>
<td>00:03</td>
<td>131764</td>
<td>/lib64/ld-2.12.so</td>
</tr>
<tr>
<td>392f420000-392f421000</td>
<td>r-wp</td>
<td>00000000</td>
<td>00:00</td>
<td>0</td>
<td>/lib64/ld-2.12.so</td>
</tr>
<tr>
<td>392f600000-392f775000</td>
<td>r-xp</td>
<td>00000000</td>
<td>08:03</td>
<td>131765</td>
<td>/lib64/libc-2.12.so</td>
</tr>
<tr>
<td>392f775000-392f975000</td>
<td>r-xp</td>
<td>00175000</td>
<td>08:03</td>
<td>131765</td>
<td>/lib64/libc-2.12.so</td>
</tr>
<tr>
<td>392f975000-392f979000</td>
<td>r-xp</td>
<td>00175000</td>
<td>08:03</td>
<td>131765</td>
<td>/lib64/libc-2.12.so</td>
</tr>
<tr>
<td>392f979000-392f97a000</td>
<td>r-xp</td>
<td>00175000</td>
<td>08:03</td>
<td>131765</td>
<td>/lib64/libc-2.12.so</td>
</tr>
<tr>
<td>392f97a000-392f97f000</td>
<td>r-xp</td>
<td>00175000</td>
<td>08:03</td>
<td>131765</td>
<td>/lib64/libc-2.12.so</td>
</tr>
<tr>
<td>7ff51f5c0000-7ff51f8b1000</td>
<td>r-xp</td>
<td>00000000</td>
<td>08:03</td>
<td>1971414</td>
<td>/usr/lib/locale/locale-archive</td>
</tr>
<tr>
<td>7ff51f8b1000-7ff51f8b4000</td>
<td>r-xp</td>
<td>00000000</td>
<td>00:00</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7ff51f8b9a000-7ff51f8b9e000</td>
<td>r-xp</td>
<td>00000000</td>
<td>00:00</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7ff80eb8000-7ff80ec5000</td>
<td>r-xp</td>
<td>00000000</td>
<td>00:00</td>
<td>0</td>
<td>[stack]</td>
</tr>
<tr>
<td>7ff80eb8000-7ff80ec5000</td>
<td>r-xp</td>
<td>00000000</td>
<td>00:00</td>
<td>0</td>
<td>[vds]</td>
</tr>
<tr>
<td>7ff80eb8000-7ff80ec5000</td>
<td>r-xp</td>
<td>00000000</td>
<td>00:00</td>
<td>0</td>
<td>[vsyscall]</td>
</tr>
<tr>
<td>rfffffffe000000-fffffff601000</td>
<td>r-xp</td>
<td>00000000</td>
<td>00:00</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Learn how your computer is using memory

- Use Linux
  - May be via live CDs

- Compare address spaces
  - 32 bit
  - 64 bit
  - Specially interesting if you have 4 GiB or more installed memory
Linux & Memory Maps