Large Virtual Address support (52-bit) in ARM64 kernel

SPEAKER
Bhupesh Sharma
Linaro
<bhupesh.sharma@linaro.org>
Twitter: @bhupesh_sharma
Part of Linaro Landing team.

Been hacking on boot loaders & kernel since past 15 years.

Contribute to:
• Linux,
• EFI/u-boot bootloader, and
• User-space utilities like:
  • kexec-tools, and
  • makedumpfile.

Co-maintain crash-utility tool
Outline

- Large VA support for arm64
  - What?
  - How?
- 52-bit VA kernel support - arm64
- Flipping the arm64 kernel memory layout
- Impact on user-space applications
- 52-bit userspace VA
- How to test 52-bit VA support
- Next Steps
Large VA support for arm64 – What?

• 64-bit hardware can address very large address space.
  – Upto 16 EiBs \((16 \times 1024^6 = 264 = 18,446,744,073,709,551,616\text{ bytes})\)
  – Approx 18.4 exabytes of memory.
• Servers available with 64 TiB (& upwards) of memory.
  – Use-cases requiring addressing spaces > \(2^{48}\) bytes.
• Limitations
  – Not all instruction sets, and
  – Not all processors,
    • support a full 64-bit virtual or physical address space.
Large VA support for arm64 – What?

- **x86_64**
  - Supports 5-level page tables in both **Hardware** & **Software**.
  - Allows addressing address space = $2^{57}$ bytes.
  - Bumps limits to
    - $\rightarrow$ 128 PiB of virtual address space,
    - $\rightarrow$ 4 PiB of physical address space.

- **arm64**
  - Introduces **2 new architecture extensions**
    - 52-bit addressing extensions
      - ARMv8.2 **LVA**, and
      - ARMv8.2 **LPA**
  - Allows addressing
    - $\rightarrow$ 4 PiB of virtual address space,
    - $\rightarrow$ 4 PiB of physical address space.

*Reference: [x86 Ice Lake processor documentation](#) & [Armv8-A architecture evolution document](#) from ARM*
Large VA support for arm64 – How?

- **ARMv8.2 LVA**
  - Supports larger VA space
  - Each translation table base register of up to 52 bits
    - when using the 64KB translation granule.
- **ARMv8.2 LPA**
  - Allows larger intermediate physical address (IPA), and
  - PA space of up to 52 bits when using the 64KB translation granule.
  - Allows a level 1 block size where the block covers a 4TB address range for the 64KB translation granule if the implementation support 52 bits of PA.

  *NOTE*: These features are supported in AArch64 state only.

- **Cortex-A processors with ARMv8.2 extension support:**
  - Cortex A55
  - Cortex A75
  - Cortex A76

*Reference*: ARMv-8 *Architecture Reference Manual* from ARM
Large VA support for arm64 – How?

- **Translating** a virtual address to a physical address

*Reference*: [Memory management guide](https://www.arm.com) from ARM
Large VA support for arm64 – How?

- arm64 Linux uses:
  - 4KB page size
    - 39-bit (512GB) virtual addresses ➔ 3 translation tables levels
    - 48-bit (256TB) virtual addresses ➔ 4 translation tables levels.

**Reference:**

Documentation/arm64/memory.rst

---

### Translation table lookup with 4KB pages:

<table>
<thead>
<tr>
<th>63</th>
<th>56</th>
<th>55</th>
<th>48</th>
<th>47</th>
<th>40</th>
<th>39</th>
<th>32</th>
<th>31</th>
<th>24</th>
<th>23</th>
<th>16</th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
|    |    |    |    |    |    |    |    |    |    |    |    | [11:0] in-page offset
|    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    | [47:39] L0 index
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | [63] TTBR0/1 |

---

**Software Support**
Large VA support for arm64 – How?

arm64 Linux uses:

- **64KB page size**
  - 42-bit (4TB) virtual addresses \( \Rightarrow 2 \) translation tables levels, but the memory layout is the same.
  - 48-bit (256TB) virtual addresses \( \Rightarrow 3 \) translation tables levels
  - 52-bit (4PiB) virtual addresses \( \Rightarrow 3 \) translation tables levels with ARMv8.2 extension
    - expands number of descriptors in the **first** level of translation.

Reference:
[Documentation/arm64/memory.rs](https://arm64/memory.rs)
Large VA support for arm64 – How?

- A sample arm64 translation table walk

Reference: ARMv8-A Address Translation from ARM
52-bit VA kernel support - arm64

• **Design problem** from a software support p-o-v
  - Older arm64 CPUs which *don’t support* ARMv8.2 extensions.
  - New / Upcoming arm64 CPUs *which support* ARMv8.2 extensions.
• **Selected design** approach
  - Have a **single** kernel binary
    • At *early boot time* check if the ARMv8.2 hardware feature is present or not.

Reference: [Documentation/arm64/memory.rst](Documentation/arm64/memory.rst)
52-bit VA kernel support - arm64

- **Single kernel binary** for both 48-bit and 52-bit VA spaces.
- **VMEMMAP** constraints
  - must be sized large enough for **52-bit VAs**, and
  - must be sized large enough to accommodate a fixed **PAGE_OFFSET**.
- **VA bits related** variables used by kernel code:

<table>
<thead>
<tr>
<th>VA_BITS</th>
<th>Compile time constant</th>
<th>Maximum size of VA space, used for things like static array and region sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>VA_BITS_MIN</td>
<td>Compile time constant</td>
<td>Minimum size of VA space, used to ensure pointers are addressable</td>
</tr>
<tr>
<td>VA_BITS_ACTUAL</td>
<td>Variable</td>
<td>The actual size of the kernel VA space</td>
</tr>
</tbody>
</table>

* vabits_actual

Reference: [Documentation/arm64/memory.rst](Documentation/arm64/memory.rst)
Flipping the kernel memory layout - arm64

Direct Linear Map is in Higher Half of the VA space.

Reference: Documentation/arm64/memory.rst

- kernel text addresses is kept constant, even for 48 to 52-bits migration.
- We need to flip the VA space.
Impact on user-space applications - arm64

- User-space applications **impacted** due to **flipped** kernel memory layout
  - used to debug running / live kernels,
  - to analyze vmcore dumps.
    - for example: **kexec-tools**, **makedumpfile** & **crash-utility**.

- Debugging applications need to perform a **va_to_pa()** conversion
  - Walk the translation table(s) for determining the physical address

- These applications are broken upstream currently.
- Have proposed fixes for affected applications
  - some have been **accepted** upstream,
  - others are still **pending**
    - **kexec-tools fix**
52-bit userspace VA - arm64

- What happens to other existing applications?

- To maintain **backward compatibility**
  - **kernel** will, by default, return virtual addresses to userspace from a **48-bit** range.

- **Opt-in** model for willing user-space application(s)
  - Hint parameter is passed to `mmap()` calls to receive addresses in 52-bit range.

```
maybe_high_address = mmap(~0UL, size, prot, flags,...);
```

- How to build kernel which returns addresses in 52-bit range?
  - Enable **CONFIG options**:
    - `CONFIG_EXPERT` && `CONFIG_ARM64_FORCE_52BIT`

*NOTE:* Only intended for debugging + should **not** be used in production.
How to test 52-bit VA support

- What if I have *no* arm64 **hardware**?
- Use **qemu**

```bash
$ virt-builder --arch aarch64 --root-password password:fedora fedora-30
```

```bash
$ qemu-system-aarch64 -no-user-config -nodefaults -display none -m 2048 -cpu cortex-a57 -machine virt -smp 4 -bios /usr/share/edk2.git/aarch64/QEMU_EFI.fd -device virtio-scsi-device,id=scsi -drive file=fedora-30.img,format=raw,if=none,id=hd0 -device scsi-hd,drive=hd0 -netdev user,id=usernet -device virtio-net-device,netdev=usernet -boot efi -serial mon:stdio
```

- Use **ARMv8 fast model** simulator for some quick checks

Reference: Fedora AArch64 Wiki
Next Steps

• Fix broken userspace applications – *WIP*.
• Create awareness about the flipped kernel address map.
• JIT and other applications need to be made aware about the `mmap()` hint parameter and how they can receive addresses in 52-bit range.
• Ard Biesheuvel has posted a patchset to extend the Linear range for 52-bit configurations:
  - https://lore.kernel.org/linux-arm-kernel/20201008153602.9467-3-ardb@kernel.org/
• Test upstream kernel on both old CPUs (48-bit VA) and new CPUs with 52-bit VA)
  - In absence of a real 52-bit HW, you can use ARMv8 fast model simulator for some quick checks.
Slides can be found on github

Email: <bhupesh.sharma@linaro.org>
Twitter: @bhupesh_sharma

The talk is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.
Questions?