Improve the container image compatibility on Arm

Wei.Chen@arm.com
Penny.Zheng@arm.com

Edinburgh, UK / Open Source Summit Europe 2018
2018-10-24
Agenda

• Background
• Why image compatibility on Arm is an issue
• What we have done to address this issue
• Next steps
• Q & A
Background

We have many user scenarios:
Background

We have many arm processors:
Background

Arm Architecture define how an ARM processor must operate

- the programmers model
- the instruction set
- system configuration
- exception handling
- the memory model

• Note that implementation of the same architecture can be very different:
  • Cortex-A8 - architecture v7-A with a 13-stage pipeline
  • Cortex-A9 - architecture v7-A with an 8-stage pipeline
Why image compatibility on Arm is an issue

![Diagram showing single-arch and multi-arch images]

Single-arch image

Multi-arch image
Manifest list

Media Types:

application/vnd.docker.distribution.manifest.list.v2
+json

Field Description:

manifests array

The manifests field contains a list of manifests for specific platforms

Requirement:

v2.2 image spec(Jan 2016)

A manifest list contains platform segregated references to single-platform manifest entries
Manifest List and Manifests

- schemaVersion
- mediaType
- manifests
  - amd64/linux
  - arm64/v8/linux
  - arm32/v7/linux
  - ppc64le/linux
  - amd64/windows

- Mediatype
- Size
- Digest
- Platform
  - architecture
  - os
  - os.version
  - os.features
  - variant
  - features

- schemaVersion
- mediaType
- config
- layers
  - layer 1
  - layer 2
  - layer 3

Manifest
Manifest List: Platform

The minimum runtime requirements of the image

- All-platform required:
  - architecture ($GOARCH)
  - os ($GOOS)

- Arm-platform required:
  - Variant

<table>
<thead>
<tr>
<th>ISA/ABI</th>
<th>architecture</th>
<th>variant</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM 32-bit, v6</td>
<td>arm</td>
<td>v6</td>
</tr>
<tr>
<td>ARM 32-bit, v7</td>
<td>arm</td>
<td>v7</td>
</tr>
<tr>
<td>ARM 32-bit, v8</td>
<td>arm</td>
<td>v8</td>
</tr>
<tr>
<td>ARM 64-bit, v8</td>
<td>arm64</td>
<td>v8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$GOOS</th>
<th>$GOARCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>android</td>
<td>arm</td>
</tr>
<tr>
<td>darwin</td>
<td>386</td>
</tr>
<tr>
<td>dragonfly</td>
<td>amd64</td>
</tr>
<tr>
<td>freebsd</td>
<td>386</td>
</tr>
<tr>
<td>linux</td>
<td>arm</td>
</tr>
<tr>
<td>linux</td>
<td>amd64</td>
</tr>
<tr>
<td>linux</td>
<td>arm</td>
</tr>
<tr>
<td>linux</td>
<td>arm64</td>
</tr>
</tbody>
</table>

© 2017 Arm Limited
Manifest-tool

A command line utility of viewing, creating, and pushing the new manifests list

- **Features:**
  - Inspect: show structure of manifest list objects
  - Create: create manifest list entries via yaml or command line arguments
  - Push: push manifest list entries to a Docker registry v2.2 API-supporting repository

- **Evolution:**
  - docker manifest subcommand (included in 18.02)
    - replace the use of manifest-tool in most scenarios
    - experimental command on the Docker client
Manifest List

```
root@ubuntu:-# manifest-tool inspect alpine:latest
Name: alpine:latest (type: application/vnd.docker.distribution.manifest.list.v2+json)
Digest: sha256:764307b3b8f5b54027bd60d07398f8d593a0918dd0d3ce30b0e93be17e430
* Contains 6 manifest references:
  1. Mfst Type: application/vnd.docker.distribution.manifest.v2+json
     Digest: sha256:8873c923e00e0fd2ba780c7b0bfb64a105e1ecb76789161f7f776311e45bf634b
     Mfst Length: 528
     Platform:
       - OS: linux
       - OS Vers: []
       - OS Feature: []
       - Arch: amd64
       - Variant:
       - Feature:
     # Layers: 1
     layer 1: digest = sha256:8e3ba11cc2a2b39ab372c60c16b42153e50e5ce64a0bc81765c2e38381bcff6

  2. Mfst Type: application/vnd.docker.distribution.manifest.v2+json
     Digest: sha256:78f33cd48ccoa55709b65c8fbd3ef81ed922c5393b064d63b0d35f51e0c9fb2e
     Mfst Length: 735
     Platform:
       - OS: linux
       - OS Vers: []
       - OS Feature: []
       - Arch: arm
       - Variant: v6
       - Feature:
     # Layers: 2
     layer 1: digest = sha256:ee7d700abbf209aa401ef5d53f86af29ba25e8154b3259036e9307d8f25b5c5d
     layer 2: digest = sha256:a1653f4092c1ccea09cd461214df3f171957f6e97a2014371dd4da10ebaec19
```
Status of multi-arch image

• Official Docker Image
• Microsoft .NET Core
• LinuxKit
Status of multi-arch image on Arm

Why image compatibility on Arm is an issue

Status of multi-arch image on Arm
First issue: containerd/containerd #1575

Error occurred in pulling multi-arch image

```
Error: Manifest not found
```

Containerd V1.0
First issue: containerd/containerd #1575

Error occurred in pulling multi-arch image

• Equal-match of architecture, os and variant

```go
func (m *matcher) Match(platform specs.Platform) bool {
    normalized := Normalize(platform)
    return m.OS == normalized.OS &&
    m.Architecture == normalized.Architecture &&
    m.Variant == normalized.Variant
}
```

• Lack of variant-detection in host platform

```go
// DefaultSpec returns the current platform's default platform specification.
func DefaultSpec() specs.Platform {
    return specs.Platform{
        OS:      runtime.GOOS,
        Architecture: runtime.GOARCH,
        // TODO(stevvooe): Need to resolve GOARM for arm hosts.
    }
}
```
CPU Identification

Different architecture has different tools to determine processor type and the presence of features

- **amd64:**
  - processor supplementary instruction: CPUID opcode

- **powerpc:**
  - 32-bit read-only PVR register

- **arm/arm64:**
  - system control register: Main ID register
Main ID Register

one of system control register, provide identification information for processor

- Attribute
  - read-only
  - 32-bit
  - privileged access

Defined by CPUID scheme

Description of the product revision status
Why /proc/cpuinfo?

• Benefit:
  • widely-available in mainstream linux kernel and varied distribution
    - e.g. ubuntu, centos, fedora, etc.
  • no additional dependencies on host OS tooling
    - e.g. lscpu

• Drawback:
  • lack of rigorous format
    - Some very old kernel would export “CPU architecture” as aarch64 not v8
/proc/cpuinfo

Obtain Arm Architecture in user space

- **CPU architecture**
  - arm
    - MRC p15, 0, <Rt>, c0, c0, 0; Read Main ID Register
  - arm64
    - Constant value: 8

```
#define read_cpuid(reg)
{
    unsigned int __val;
    asm("mrc p15, 0, %0, c0, c0, " __stringify(reg) :
        ":=r" __val
        " : "cc");
    __val;
}
```

```
root@ubuntu:~ # cat /proc/cpuinfo
processor : 0
BogoMIPS   : 400.00
Features   : fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics cpuid asimd rdm
CPU implementer : 0x43
CPU architecture: 8
CPU variant   : 0x1
CPU part      : 0x0af
CPU revision  : 0
```

```c
seq_printf(m, "CPU implementer\t: 0x%02x\n", MIDR_IMPLEMENTOR(mdr));
seq_printf(m, "CPU architecture: 8\n");
```
Human-readable variant (OCI spec)

Refine ‘CPU Architecture’ into human-readable format

variant in /proc/cpuinfo:

```c
static const char *proc_arch[] = {
    "undefined/unknown",
    "3",
    "4",
    "4T",
    "5",
    "5T",
    "5TE",
    "5TEJ",
    "6TEJ",
    "7",
    "7M",
    "?(12)",
    "?(13)",
    "?(14)",
    "?(15)",
    "?(16)",
    "?(17)",
};
```

variant in OCI spec:

<table>
<thead>
<tr>
<th>CPU Architecture</th>
<th>Human-readable Variant</th>
<th>Simplified Variant</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Armv3</td>
<td>v3</td>
</tr>
<tr>
<td>4</td>
<td>Armv4</td>
<td>v4</td>
</tr>
<tr>
<td>4T</td>
<td>Armv4 with Thumb instruction set</td>
<td>v4</td>
</tr>
<tr>
<td>5</td>
<td>Armv5</td>
<td>v5</td>
</tr>
<tr>
<td>5T</td>
<td>Armv5 with Thumb instruction set</td>
<td>v5</td>
</tr>
<tr>
<td>5TE</td>
<td>Armv5T with DSP extensions</td>
<td>v5</td>
</tr>
<tr>
<td>5TEJ</td>
<td>Armv5TE with Jazelle* extensions</td>
<td>v5</td>
</tr>
<tr>
<td>6TEJ</td>
<td>Armv6TE with Jazelle* extensions</td>
<td>v6</td>
</tr>
<tr>
<td>7</td>
<td>Armv7</td>
<td>v7</td>
</tr>
<tr>
<td>7M</td>
<td>Armv7:Thumb-2 instructions only</td>
<td>v7</td>
</tr>
<tr>
<td>8</td>
<td>Armv8</td>
<td>v8</td>
</tr>
</tbody>
</table>

......
**Special Case**

Confusion between Armv6 and Armv7

Confusion output on core ARM1176JZF-S
Special Case
Confusion between Armv6 and Armv7

- **ARM1176JZF-S:**
  - armv6 but have support for *Virtual Memory System Architecture* (VMSA) v7

- Selection criteria in `/proc/cpuinfo`:
  - code-checking whether it supports *Virtual Memory System Architecture* (VMSA) v7 or *Protected Memory System Architecture* (PMSA) v7
  - extra check for armv7 (on discussion)
    - e.g. `uname -m`

```c
/* Revised CPUID format. Read the Memory Model Feature * Register 0 and check for VMSAv7 or PMSAv7 */
unsigned int mmfr0 = read_cpubid_ext(CPUID_EXT_MMFRO);
if (((mmfr0 & 0x0000000f) == 0x00000003 ||
     (mmfr0 & 0x000000f0) == 0x00000030))
    cpu_arch = CPU_ARCH_ARMv7;
else if (((mmfr0 & 0x0000000f) == 0x00000002 ||
         (mmfr0 & 0x000000f0) == 0x00000020))
    cpu_arch = CPU_ARCH_ARMv6;
else
    cpu_arch = CPU_ARCH_UNKNOWN;
```
Issue: moby/moby #36121

Add variant-matching scheme for arm host

Add multi-arch image support for ARM #36121

Pennyzct wants to merge 1 commit into moby:master from Pennyzct:multi-arch

Conversation 20 Commits 1 Checks 0 Files changed 2

Pennyzct commented on Jan 26 • edited

When pulling multi-arch image from official repo, mismatch variant may be pulled, due to now only arch and os will be verified. Relevant issue has already been raised in github (#34875).
Not long ago, I worked with wei @Weichen81 on multi-arch image support in containerd, and the relevant code has already be merged (containerd/containerd#1575). I reused logic and code from that and modified some based on docker infrastructure.

@arm64b
Failure Mode

Fail on exact variant-matching

- exact variant match doesn't exist:
  - e.g. hardware = v7, and there is no v7 image in the registry, only v6 (failed)

- backward compatibility:
  - arch: arm
    - armv7 -> armv6 -> armv5
  - armv6 -> armv5
  - arch: arm64 (only v8)
    - to be continued
# Next steps

Status of variant-matching code on mainstream container runtime

<table>
<thead>
<tr>
<th></th>
<th>containerd</th>
<th>docker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variant Detection</td>
<td>merged</td>
<td>under review</td>
</tr>
<tr>
<td>Variant Matching</td>
<td>merged</td>
<td>under review</td>
</tr>
<tr>
<td>Failure Mode</td>
<td>merged</td>
<td>under review</td>
</tr>
<tr>
<td>Extra check between armv7 and armv6</td>
<td>under discussion</td>
<td>under discussion</td>
</tr>
</tbody>
</table>

containerd/containerd: 1. Fix pull-multi-arch images for Arm [#1575](#1575) 2. Add platform match comparer interface [#2581](#2581)

moby/moby: 1. Add multi-arch image support for ARM [#36121](#36121)
Thank You!
Danke!
Merci!
谢谢!
ありがとうございます!
Gracias!
Kiitos!