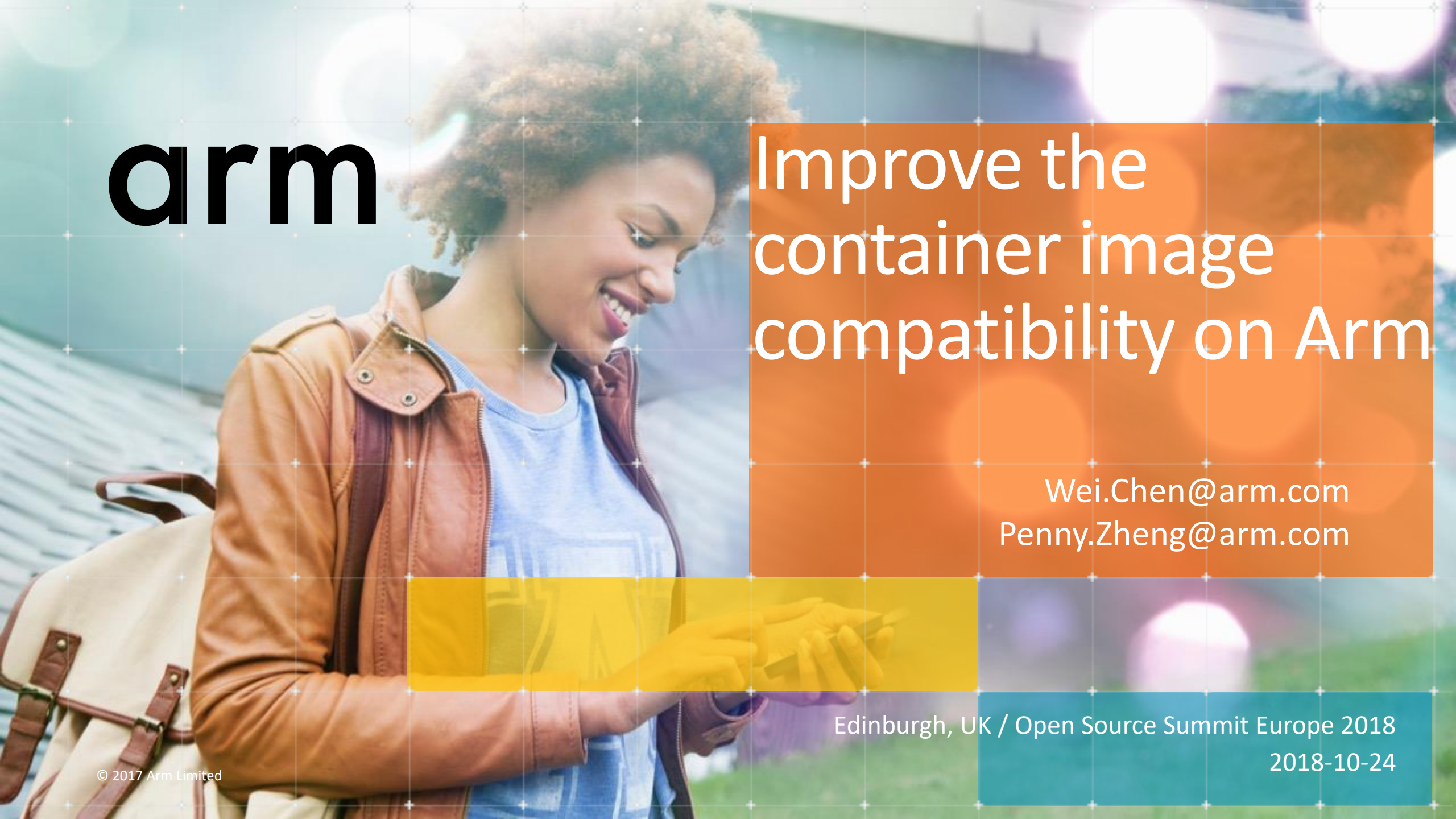


Improve the container image compatibility on Arm

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

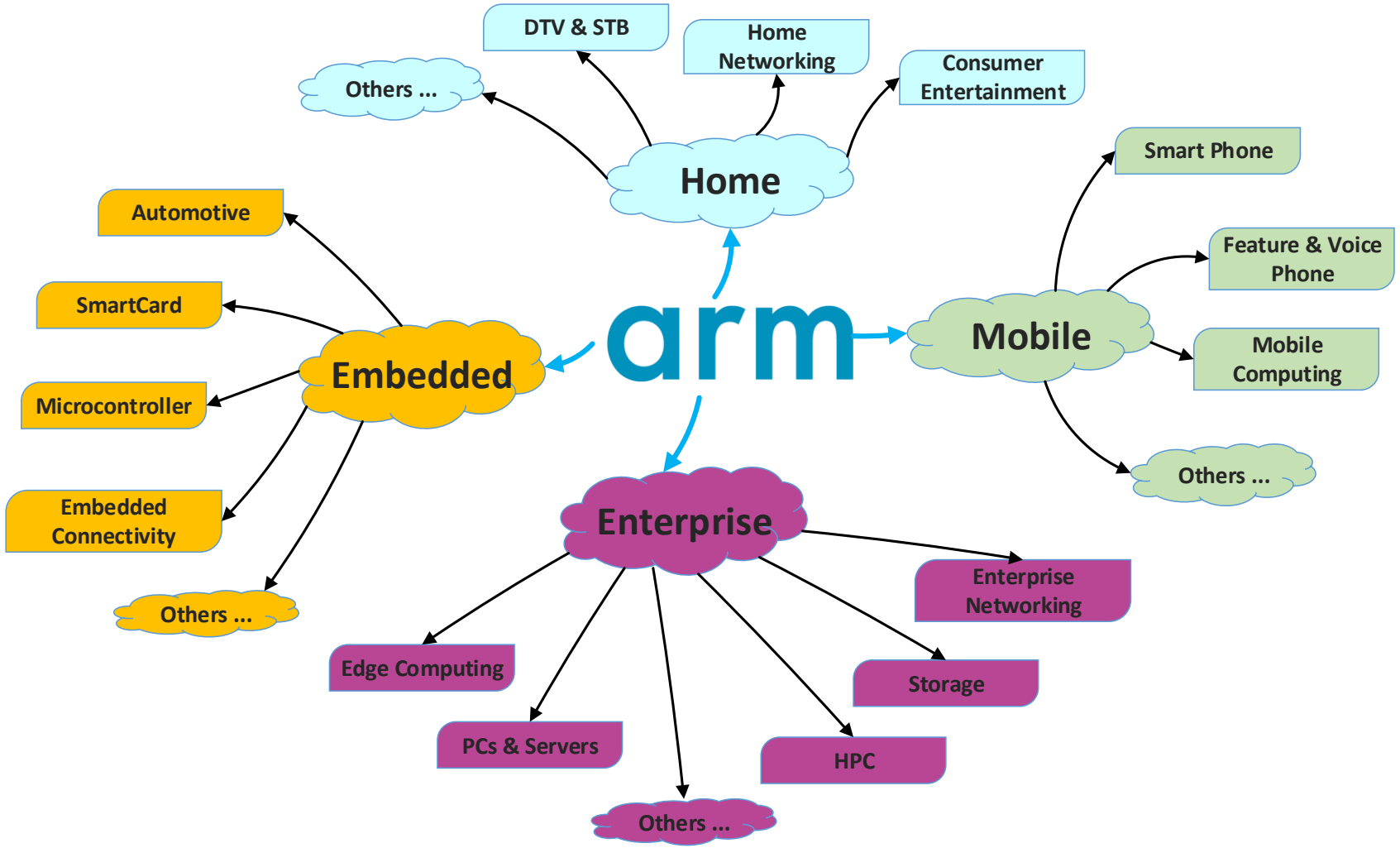
Edinburgh, UK / Open Source Summit Europe 2018
2018-10-24

Agenda

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

Background

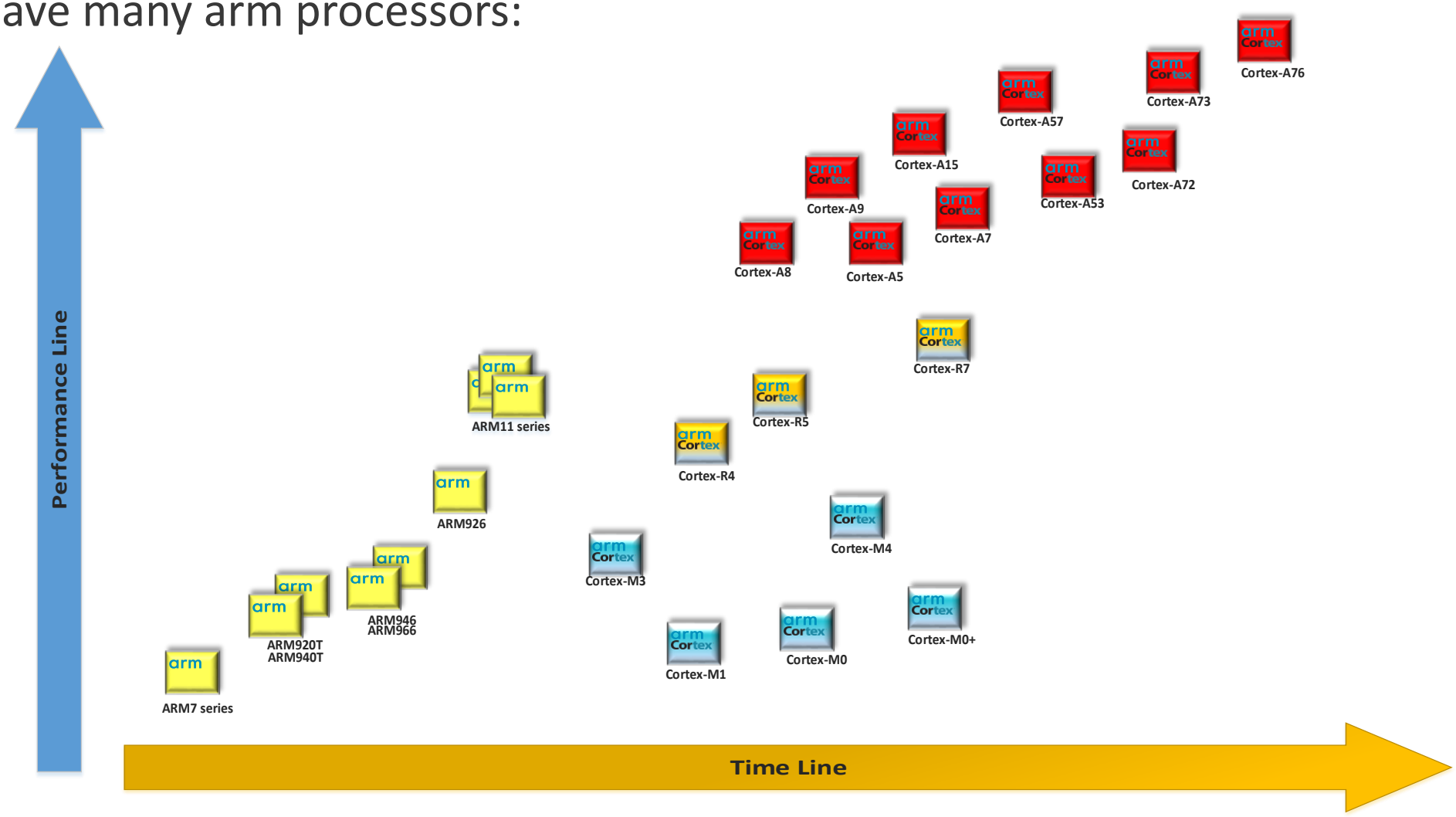
We have many user scenarios:



arm

Background

We have many arm processors:



Background

Arm Architecture define how an ARM processor must operate

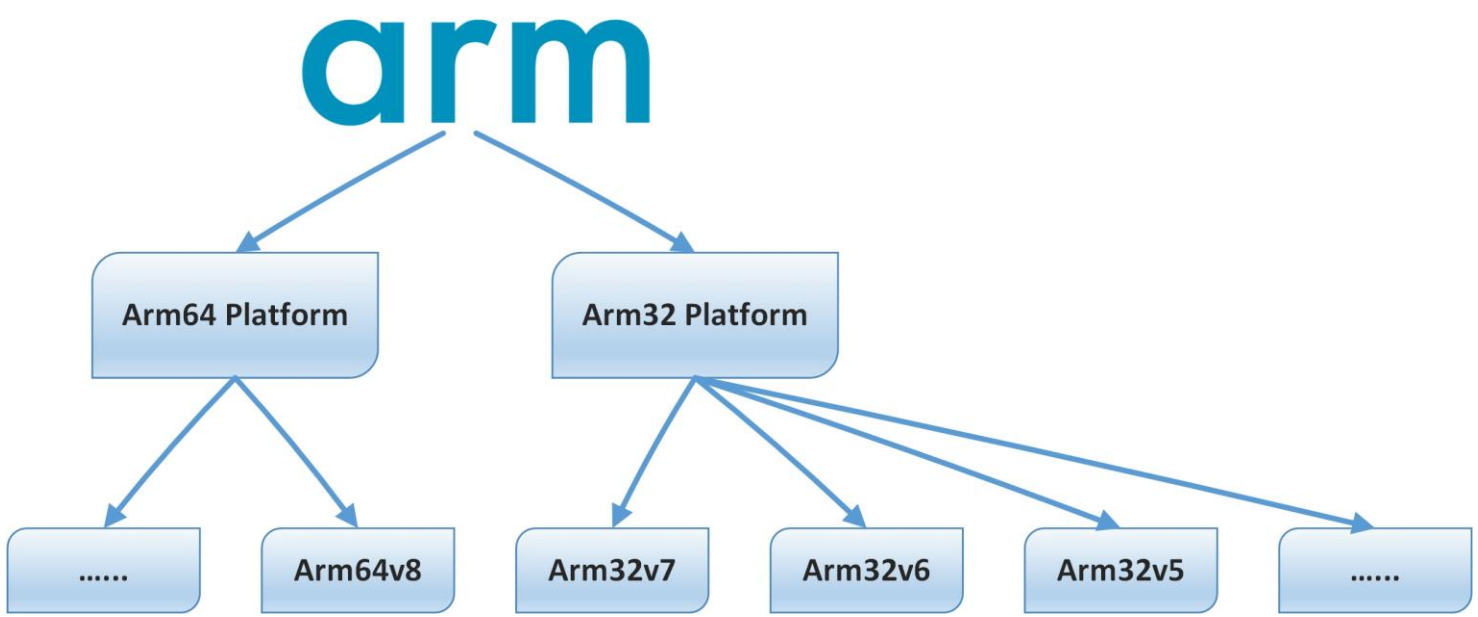
- the programmers model
- the instruction set
- system configuration
- exception handling
- the memory model

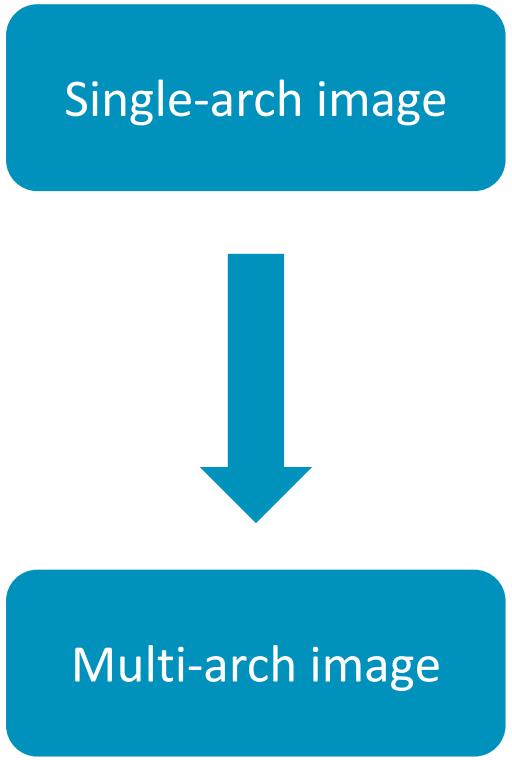
Arm Core	Arm Architecture
ARM1136J(F)-S	ARMv6
Cortex-M0	ARMv6-M
Cortex-A5	ARMv7-A
Cortex-A8	
Cortex-A9	
Cortex-A15	
Cortex-A17	
Cortex-A53	ARMv8-A
Cortex-A57	
Cortex-A72	
Cortex-A73	
Cortex-A55	ARMv8.2-A
Cortex-A75	
……	

- 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

arm

Why image compatibility on Arm is an issue





arm

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

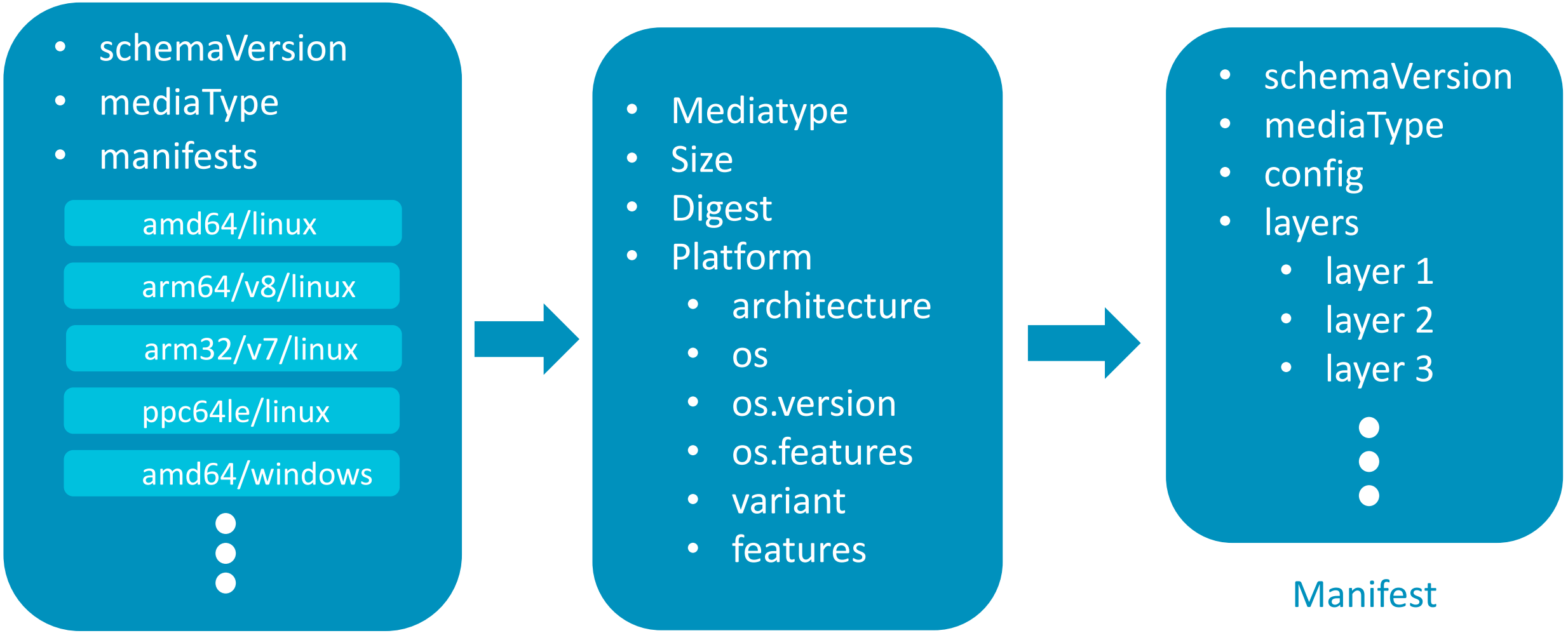
Requirement:

v2.2 image spec(Jan 2016)

A manifest list contains platform segregated references to single-platform manifest entries

arm

Manifest List and Manifests



arm

Manifest List: Platform

The minimum runtime requirements of the image

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

ISA/ABI	`architecture`	`variant`
ARM 32-bit, v6	`arm`	`v6`
ARM 32-bit, v7	`arm`	`v7`
ARM 32-bit, v8	`arm`	`v8`
ARM 64-bit, v8	`arm64`	`v8`

$GOOS	$GOARCH
android	arm
darwin	386
darwin	amd64
darwin	arm
darwin	arm64
dragonfly	amd64
freebsd	386
freebsd	amd64
freebsd	arm
linux	386
linux	amd64
linux	arm
linux	arm64

●●●

arm

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

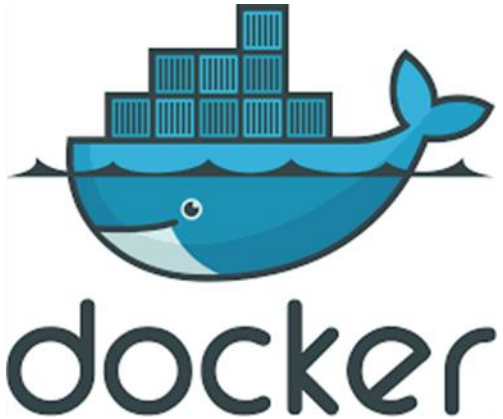
```
root@ubuntu:~# manifest-tool inspect alpine:latest
Name:   alpine:latest (Type: application/vnd.docker.distribution.manifest.list.v2+json)
Digest: sha256:7043076348bf5040220df6ad703798fd8593a0918d06d3ce30c6c93be117e430
 * Contains 6 manifest references:
1    Mfst Type: application/vnd.docker.distribution.manifest.v2+json
1       Digest: sha256:0873c923e00e0fd2ba78041bfb64a105e1ecb7678916d1f7776311e45bf5634b
1  Mfst Length: 528
1     Platform:
1           -      OS: linux
1           - OS Vers:
1           - OS Feat: []
1           -    Arch: amd64
1           - Variant:
1           - Feature:
1     # Layers: 1
         layer 1: digest = sha256:8e3ba11ec2a2b39ab372c60c16b421536e50e5ce64a0bc81765c2e38381bcff6

2    Mfst Type: application/vnd.docker.distribution.manifest.v2+json
2       Digest: sha256:78f3ccd48cc6a55709b65c8fdb3ef81ed922c5393b064d63b0d35f51e0c9fb2e
2  Mfst Length: 735
2     Platform:
2           -      OS: linux
2           - OS Vers:
2           - OS Feat: []
2           -    Arch: arm
2           - Variant: v6
2           - Feature:
2     # Layers: 2
         layer 1: digest = sha256:ee7d700abbf209aa401ef5d53f86af298a25e8154b3259036e9307d08f255c5d
         layer 2: digest = sha256:a1653f4692c1ccea69cd46121d4f1371957f66e97a20141371dd4da10ebaec19
```


arm

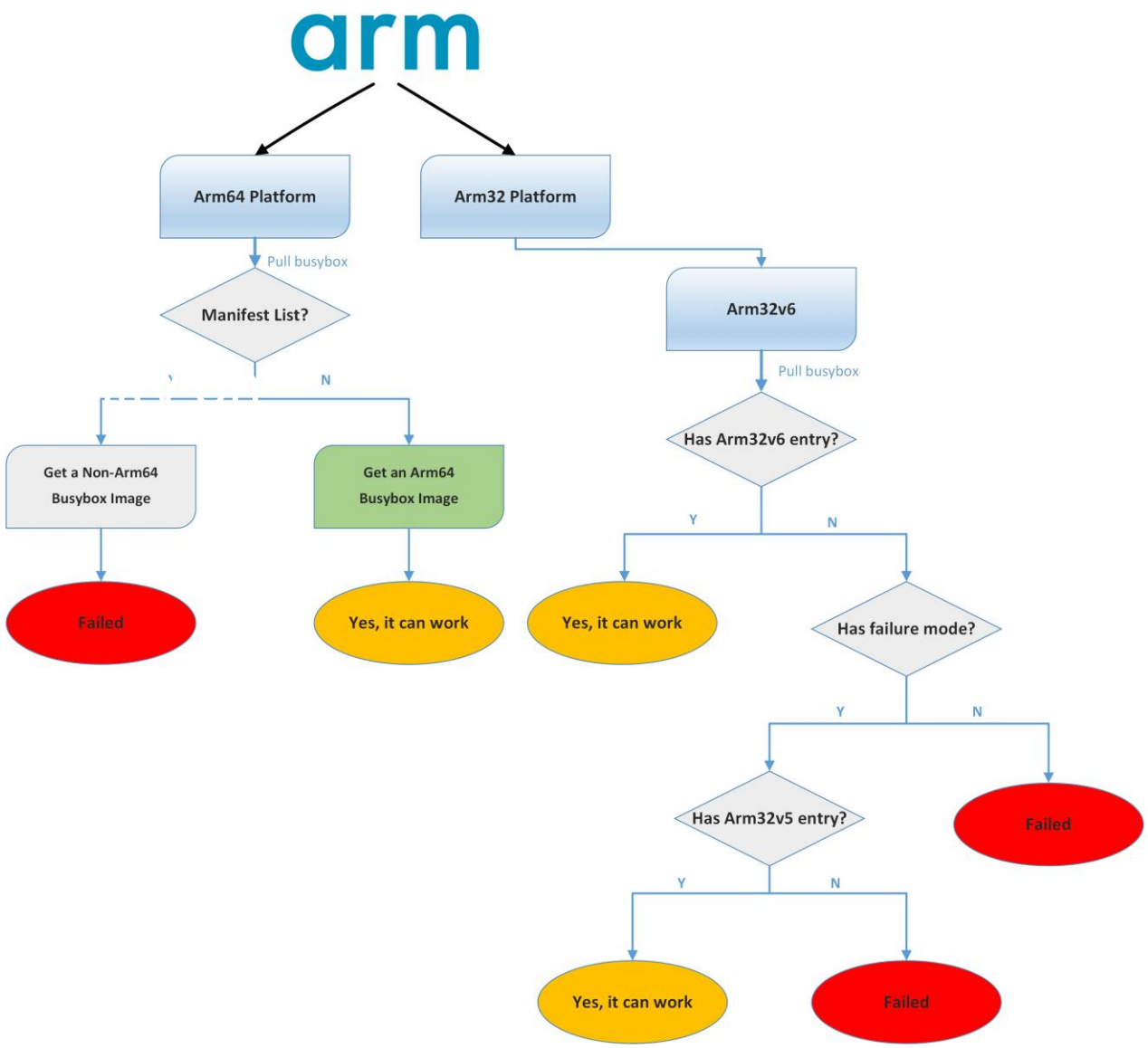
Status of multi-arch image

- Official Docker Image
- Microsoft .NET Core
- LinuxKit



arm

Status of multi-arch image on Arm



arm

First issue: containerd/containerd #1575

Error occurred in pulling multi-arch image

arm

First issue: containerd/containerd #1575

Error occurred in pulling multi-arch image

- Equal-match of architecture, os and variant

```
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

```
// 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.
        }
}
```


arm

CPU Identification

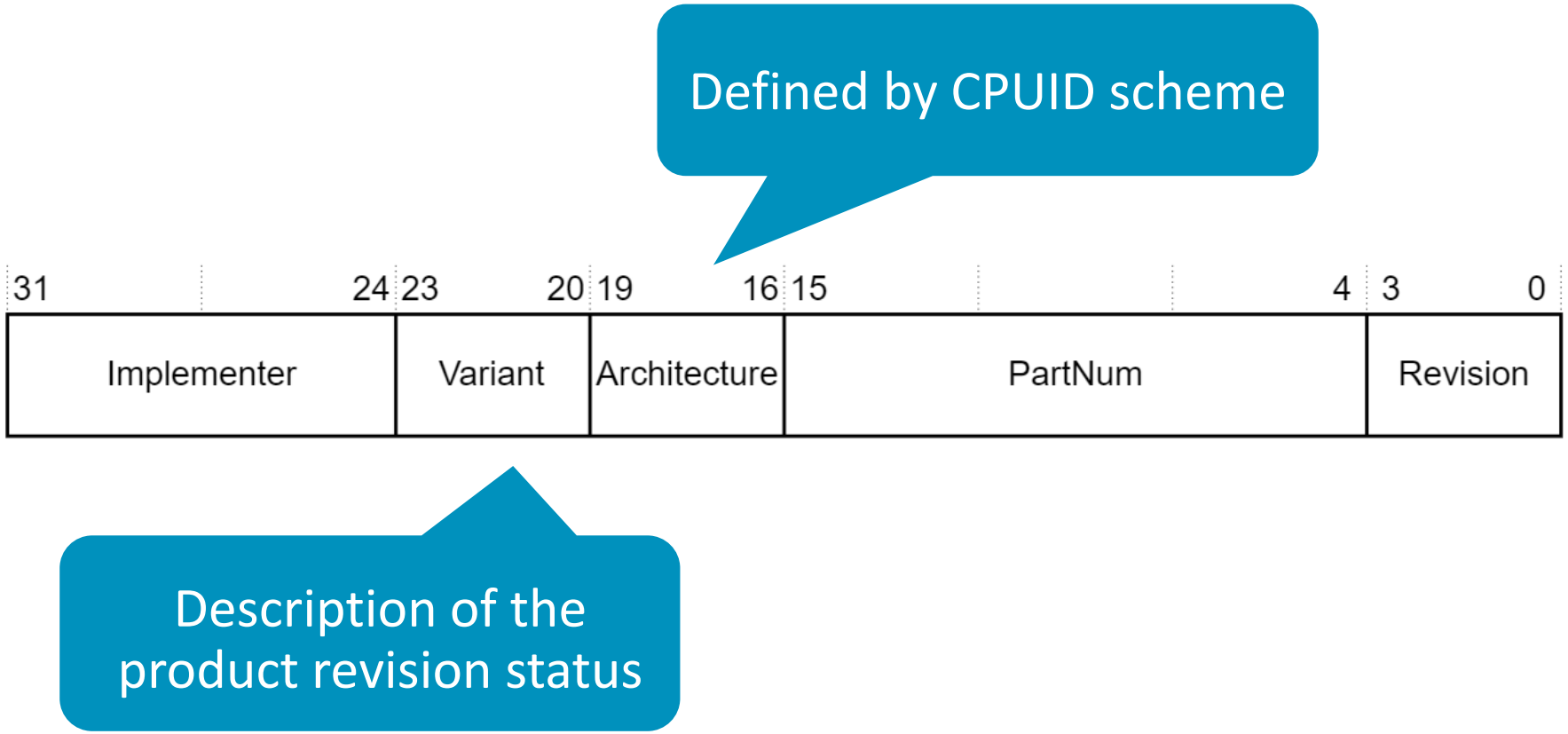
Different architecture has different tools to determine processor type and the presence of features

- 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



arm

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;
        })
```

```
seq_printf(m, "CPU implementer\t: 0x%02x\n",
           MIDR_IMPLEMENTOR(midr));
seq_printf(m, "CPU architecture: 8\n");
seq_printf(m, "CPU variant\t: 0x%x\n", MIDR_VARIANT(midr));
seq_printf(m, "CPU part\t: 0x%03x\n", MIDR_PARTNUM(midr));
seq_printf(m, "CPU revision\t: %d\n\n", MIDR_REVISION(midr));
```

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


arm

Human-readable variant (OCI spec)

Refine ‘CPU Architecture’ into human-readable format

variant in /proc/cpuinfo:

```
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:

CPU Architecture	Human-readable Variant	Simplified Variant
3	Armv3	v3
4	Armv4	v4
4T	Armv4 with Thumb instruction set	
5	Armv5	v5
5T	Armv5 with Thumb instruction set	
5TE	Armv5T with DSP extensions	
5TEJ	Armv5TE with Jazelle® extensions	
6TEJ	Armv6TE with Jazelle® extensions	v6
7	Armv7	v7
7M	Armv7:Thumb-2 instructions only	
8	Armv8	v8
.....		

arm

Special Case

Confusion between Armv6 and Armv7

AmedeeBulle commented on Jun 19

FWIW this fix does not address the Raspberry Pi case as the "Zero" has an ARMv6 chip but reports a CPU arch of 7!

```
$ arch
armv6l
$ cat /proc/cpuinfo | grep 'CPU architecture'
CPU architecture: 7
```

Confusion output on core ARM1176JZF-S

arm

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

```
/* Revised CPUID format. Read the Memory Model Feature
 * Register 0 and check for VMSAv7 or PMSAv7 */
unsigned int mmfr0 = read_cpuid_ext(CPUID_EXT_MMFR0);
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;
```


arm

Issue: moby/moby #36121

Add variant-matching scheme for arm host

Add multi-arch image support for ARM #36121

Open **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 for 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

arm

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

```
if platform.Architecture == "arm" {
        if platform.Variant == "v7" {
                return orderedPlatformComparer{
                        matchers: []Matcher{
                                &matcher{
                                        Platform: platform,
                                },
                                &matcher{
                                        Platform: specs.Platform{
                                                Architecture: platform.Architecture,
                                                OS:           platform.OS,
                                                OSVersion:    platform.OSVersion,
                                                OSFeatures:   platform.OSFeatures,
                                                Variant:      "v6",
                                        },
                                },
                                &matcher{
                                        Platform: specs.Platform{
                                                Architecture: platform.Architecture,
                                                OS:           platform.OS,
                                                OSVersion:    platform.OSVersion,
                                                OSFeatures:   platform.OSFeatures,
                                                Variant:      "v5",
                                        },
                                },
                        },
                }
}
```


arm

Next steps

Status of variant-matching code on mainstream container runtime

	containerd	docker
Variant Detection	merged	under review
Variant Matching	merged	under review
Failure Mode	merged	under review
Extra check between armv7 and armv6	under discussion	under discussion

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

moby/moby: 1. Add multi-arch image support for ARM #36121

arm

Thank You!
Danke!
Merci!
谢谢!
ありがとう!
Gracias!
Kiitos!