Using linux-yocto + yocto kernel tools to create and maintain a BSP

Bruce Ashfield, Xilinx

Yocto Project Summit, 2021.11

Introduction

Agenda

- **Background**
  - Configuration abstraction
  - Tools
- **Creating / Extending a BSP**
  - with or without kernel metadata
- **Future work**

Goals

- **Highlight configuration and BSP challenges**
  - Less about the kernel source or a specific kernel tree
- **Introduce the configuration data and tools**
  - Demonstrate (simple) creation and extension of BSPs

Overview / Background

Overview: linux-yocto

- **Multi-architecture, maintained reference**
  - Userspace / kernel stacks
  - Identical to mainline linux in many branches
- **Drives support tightly coupled kernel packages**
  - libc-headers, perf, lttng, systemtap, etc
- **Collaboration / Leverage collective testing**
  - Support BSP and assist upstreaming
- **Kernel configuration meta-data and tooling**
  - More important than you think!

Overview: linux-yocto + kern-tools

- **This talk:**
  - focus on kernel configuration management
- **Developed over 10 years of maintaining distro kernels**
  - Yes, almost everything has been tried. No, we can't equally maintain it with directories full of patches
  - Supports many different workflows
- **Scales to maintain ~6 kernel versions, 6 architectures, 14 boards (in oe-core/meta-yocto-bsp alone), 3 kernel types**
  - 30+ invasive / non-invasive features
  - Hundreds of patches at any given time

Overview: kernel configuration

- **Abstract the details from the integrator**
  - avoid kernel version specific knowledge
- **Support dynamic feature addition/removal**
  - appropriate for heterogeneous platforms or addon boards
- **Avoid “just-in-case” configurations**
  - flexible approach for large software stacks
- **Split policy from hardware configuration**

Overview: kernel configuration

- **kernel meta-data (configuration fragments)**
  - .scc/.cfg fragments
    - good: recipe/layer specific
    - better: collected in a central location
- **Integration with the build (KERNEL_FEATURES, etc)**
  - Allows mapping of distro/image features to abstracted kernel requirements
  - Also useful for machine features (if careful)

Overview: kernel-cache

- **kernel-cache fragment collection:**
  - can be used without linux-yocto
  - can be used without adding the repository to SRC_URI
    - https://git.yoctoproject.org/meta-virtualization/tree/recipes-kernel/linux/yocto-cfg-fragments.bb
  - can be overridden via layers

Overview: reference BSP configuration

- **defconfigs**
  - are a great resource
  - a good starting point (or finishing point)
  - are often “demo” configs, with a bit of everything
  - suitability depends on requirements and amount of churn
- **kernel-yocto reference BSPs use configuration fragments**
  - BSP + base kernel type + features

Creating / Extending a BSP

Creating or extending a BSP

- **Many options, but generally:**
  a. “choose” the kernel tree
  b. select the configuration (probably a defconfig)
  c. build, boot & test
  d. modify the configuration
     - performance ? footprint ? features ?
     - repeat b-c-d until satisfied/”happy”
  e. finalize configuration and machine definition

Creating a BSP: with yocto kernel tools (+linux-yocto)

- **Extending an existing linux-yocto BSP**
  - Add a configuration fragment via SRC_URI or KERNEL_FEATURES
    - Not so interesting for this presentation …. moving on
- **From a reference defconfig**
  - build and boot test
  - add feature fragments
  - split hardware from policy options

Creating a BSP from a defconfig

- Consider the following -stable recipe:

```
SUMMARY = "Linux kernel"
SECTION = "kernel"
LICENSE = "GPLv2"

inherit kernel
inherit kernel-yocto
LIC_FILES_CHKSUM = "file://COPYING;md5=6bc538ed5bd9a7fc9398086aedcd7e46"

KBRANCH = "v5.14/base"
KCONFIG_MODE="alldefconfig"
KBUILD_DEFCONFIG:qemuarm64="defconfig"
KERNEL_DANGLING_FEATURES_WARN_ONLY="t"

SRC_URI = "git://git.yoctoproject.org/linux-yocto.git;name=machine;branch=${KBRANCH}"

LINUX_VERSION ?= "5.14.18"
PV = "${LINUX_VERSION}+git${SRCPV}"

DEPENDS += "${@bb.utils.contains('ARCH', 'x86', 'elfutils-native', '', d)}"
DEPENDS += "openssl-native util-linux-native gmp-native"

SRCREV_machine ?= "a0265dd8262de73457aaa3ce1c5938dc152b8085"
SRCREV_meta ?= "7b9ba93dfc39efa90056eed0b572e86909127aaa"

COMPATIBLE_MACHINE = "(qemux86|qemux86-64|qemuarm|qemuarm64)"
```

Creating a BSP from a defconfig

- add layer, setup MACHINE, preferred kernel provider

```
% bitbake linux-stable
```

```
% runqemu qemuarm64 nographic slirp

…

qemuarm64 login: root
root@qemuarm64:~# uname -a
Linux qemuarm64 5.14.18 #1 SMP PREEMPT Fri Nov 12 14:02:57 UTC 2021
aarch64 aarch64 aarch64 GNU/Linux
root@qemuarm64:~#
```

Creating a BSP: splitting the defconfig

- **New tool: kgit-config**
  - Leverages existing kernel config meta-data
    - or basic configuration from kernel-source
  - Can: split fragments (or defconfigs) into h/w and policy
  - Can: create a skeleton BSP definition
  - Can: generate a report about available policy fragments
  - Can: query a kernel-cache for fragments and options
  - Can't: perfectly create a working BSP every time ...

Creating a BSP: splitting a defconfig

- **Type 1:**
  - no kernel-cache
- **Type 2:**
  - with kernel-cache
    - Get feature suggestions for non-hardware fragments

Creating a BSP: splitting a defconfig: type 1

```
% mkdir scratch; cd scratch
% kgit-config create --bsp mybsp --ksrc ~/poky-kernel/linux-yocto.git/ --defconfig
~/poky-kernel/linux-yocto.git/arch/arm64/configs/defconfig  -o .

kernel-cache
├── bsp
│   │
│   └── mybsp
│       ├── mybsp-hw.cfg
│       ├── mybsp-policy.cfg
│       └── mybsp.scc
└── ktypes
    └── base
        ├── basehw.cfg
        ├── basenonhw.cfg
        ├── base.scc
        ├── hardware.kcf
        └── non-hardware.kcf
```

Creating a BSP: splitting a defconfig: type 1

```
% cat kernel-cache/bsp/mybsp/mybsp.scc
# SPDX-License-Identifier: MIT
define KMACHINE mybsp
define KTYPE standard

include ktypes/base/base.scc
kconf hardware mybsp-hw.cfg
# kconf non-hardware mybsp-policy.cfg

% cat ~/poky-kernel/linux-yocto.git/arch/arm64/configs/defconfig  | wc -l
1215
% cat kernel-cache/bsp/mybsp/mybsp-hw.cfg | wc -l
1078
% cat kernel-cache/bsp/mybsp/mybsp-policy.cfg | wc -l
137
```

Creating a BSP: splitting a defconfig: type 1

- **Copy generated kernel-cache to your local layer**
  - recipes-kernel/linux-stable
- **Modify the linux-stable recipe and boot (*):**

```
% diff -u linux-stable_5.14.bb.defconfig linux-stable_5.14.bb.local.kernel-cache
--- linux-stable_5.14.bb.defconfig      2021-11-28 14:29:51.273974246 -0500
+++ linux-stable_5.14.bb.local.kernel-cache     2021-11-28 14:29:39.153339968 -0500
@@ -9,10 +9,11 @@

 KBRANCH = "v5.14/base"
 KCONFIG_MODE="alldefconfig"
-KBUILD_DEFCONFIG:qemuarm64="defconfig"
 KERNEL_DANGLING_FEATURES_WARN_ONLY="t"

 SRC_URI = "git://git.yoctoproject.org/linux-yocto.git;name=machine;branch=${KBRANCH}"
+SRC_URI += "file://kernel-cache;type=kmeta;destsuffix=kernel-cache"
+KMACHINE:qemuarm64="mybsp"
```

Creating a BSP: splitting a defconfig: type 2

```
% kgit-config create --bsp bspplusmeta --ksrc ~/poky-kernel/linux-yocto.git/ --defconfig
~/poky-kernel/linux-yocto.git/arch/arm64/configs/defconfig --kmeta ~/poky-kernel/kernel-cache -o .

# type1 has 1078 h/w options
% cat kernel-cache/bsp/bspplusmeta/bspplusmeta-hw.cfg | wc -l
1054
```

```
build [/home/bruc...ls/scratch]> diff -u
kernel-cache/bsp/mybsp/mybsp-hw.cfg
kernel-cache/bsp/bspplusmeta/bspplusmeta-hw.cfg |grep \^-
--- kernel-cache/bsp/mybsp/mybsp-hw.cfg 2021-11-26
16:53:36.334163228 -0500
-CONFIG_BT_HCIBTUSB=m
-CONFIG_BT_HCIUART=m
-CONFIG_BT_HCIUART_LL=y
-CONFIG_BT_HCIUART_BCM=y
-CONFIG_BT_HCIUART_QCA=y
-CONFIG_DEVTMPFS=y
-CONFIG_DEVTMPFS_MOUNT=y
-CONFIG_BLK_DEV_LOOP=y
-CONFIG_BLK_DEV_NBD=m
-CONFIG_MD=y
-CONFIG_BLK_DEV_MD=m
-CONFIG_BLK_DEV_DM=m
-CONFIG_DM_MIRROR=m
```

```
-CONFIG_DM_ZERO=m
-CONFIG_NETDEVICES=y
-CONFIG_MACVLAN=m
-CONFIG_MACVTAP=m
-CONFIG_TUN=y
-CONFIG_VETH=m
-CONFIG_VIRTIO_NET=y
-CONFIG_LEGACY_PTY_COUNT=16
-CONFIG_SERIAL_OF_PLATFORM=y
-CONFIG_WATCHDOG=y
-CONFIG_MAGIC_SYSRQ=y
```

Creating a BSP: splitting a defconfig: type 2

```
% kgit-config create --ksrc ~/poky-kernel/linux-yocto.git/ --defconfig ~/poky-kernel/linux-yocto.git/arch/arm64/configs/defconfig --kmeta
~/poky-kernel/kernel-cache -o .

[INFO]: BSP h/w configuration values:
CONFIG_ARCH_ACTIONS=y
CONFIG_ARCH_AGILEX=y
CONFIG_ARCH_N5X=y
CONFIG_ARCH_SUNXI=y
<....>
[INFO]: BSP policy options and fragments that provide them:
CONFIG_SYSVIPC=y (['configs/v5.14/ktypes/base/base.cfg'])
CONFIG_POSIX_MQUEUE=y (['configs/v5.14/ktypes/base/base.cfg'])
CONFIG_AUDIT=y (['configs/v5.14/ktypes/base/base.cfg'])
CONFIG_NO_HZ_IDLE=y (no fragment)
CONFIG_HIGH_RES_TIMERS=y (['configs/v5.14/standard/features/hrt/hrt.cfg'])
CONFIG_PREEMPT=y (['configs/v5.14/ktypes/base/base.cfg'])
CONFIG_IRQ_TIME_ACCOUNTING=y (no fragment)
CONFIG_BSD_PROCESS_ACCT=y (['configs/v5.14/ktypes/base/base.cfg'])
CONFIG_BSD_PROCESS_ACCT_V3=y (['configs/v5.14/ktypes/base/base.cfg'])
CONFIG_TASK_XACCT=y (no fragment)
CONFIG_TASK_IO_ACCOUNTING=y (no fragment)
CONFIG_IKCONFIG=y (['configs/v5.14/standard/ktypes/standard/standard.cfg'])
CONFIG_IKCONFIG_PROC=y (['configs/v5.14/standard/ktypes/standard/standard.cfg'])
CONFIG_NUMA_BALANCING=y (no fragment)
CONFIG_MEMCG=y (['configs/v5.14/standard/features/cgroups/cgroups.cfg'])
CONFIG_MEMCG_SWAP=y (['configs/v5.14/standard/features/cgroups/cgroups.cfg'])
```

kgit-config: query

- **Saves some grepping**
  - supports the feature suggestions
  - more features in the future

```
% kgit-config query --kmeta ~/poky-kernel/kernel-cache CONFIG_DEBUG_FS
[INFO]: looking for ['CONFIG_DEBUG_FS']
[INFO]: cfg files that have options matching regex: CONFIG_DEBUG_FS
[INFO]: 7 cfg files have option: CONFIG_DEBUG_FS
  cfg file: cfg/fs/debugfs.cfg
     included by: /home/bruce/poky-kernel/kernel-cache/cfg/fs/debugfs.scc
  cfg file: cfg/debug/irq/debug-generic-irq-debugfs.cfg
     included by: /home/bruce/poky-kernel/kernel-cache/cfg/debug/irq/debug-generic-irq-debugfs.scc
  cfg file: cfg/debug/irq/debug-irq-domain.cfg
     included by: /home/bruce/poky-kernel/kernel-cache/cfg/debug/irq/debug-irq-domain.scc
  cfg file: cfg/debug/printk/debug-dynamic-debug.cfg
     included by: /home/bruce/poky-kernel/kernel-cache/cfg/debug/printk/debug-dynamic-debug.scc
  cfg file: features/systemtap/systemtap.cfg
     included by: /home/bruce/poky-kernel/kernel-cache/features/systemtap/systemtap.scc
  cfg file: bsp/beaglebone/beaglebone-non_hardware.cfg
     included by: /home/bruce/poky-kernel/kernel-cache/bsp/beaglebone/beaglebone.scc
  cfg file: bsp/intel-x86/intel-x86-acpi.cfg
     included by: /home/bruce/poky-kernel/kernel-cache/bsp/intel-x86/intel-x86.scc
```

Future Work

- **Improve automatic splitting heuristics**
- **Feature fragments**
  - Allow the creation of feature fragments from starting option
  - Extend query and discovery capabilities
- **Consolidate more options in central repository**

yocto PROJECT | THE LINUX FOUNDATION