FLOP: CUSTOMIZING YOCTO PROJECT FOR MVMExxxx POWERPC AND BEAGLEBONE ARM

L. Pivetta, A.I. Bogani, R. Passuello
Elettra Sincrotrone Trieste, Trieste, Italy

Abstract

During the last fifteen years several PowerPC-based VME single board computers, belonging to the MVMExxxx family, have been used for the control system front-end computers at Elettra Sincrotrone Trieste. Moreover, a low cost embedded board has been recently adopted to fulfill the control requirements of distributed instrumentation. These facts lead to the necessity of managing several releases of the operating system, kernel and libraries, and finally to the decision of adopting a comprehensive unified approach based on a common codebase: the Yocto Project. Based on Yocto Project, a control system oriented GNU/Linux distribution called Flop has been created. The complete management of the software chain, the ease of upgrading or downgrading complete systems, the centralized management and the platform-independent deployment of the user software are the main features of Flop.

INTRODUCTION

Since the last decades, technology evolution and component obsolescence are driving a renewal that involves also particle accelerator control system platforms. Even industrial grade products well-known for the long lifetime support, such as VME, suffer these issues. Moreover, the evolution of application software, especially when dealing with new compiler features or when requiring new system library functionalities, often involves an update of the operating system software. Several models of PowerPC-based VME single board computers are in use at Elettra, running different distributions/releases of the GNU/Linux operating system [1–3]. Furthermore, the BeagleBone, a low cost embedded board based on the ARM microprocessor, has been recently introduced, as well as a low-power high-performance INTEL Bay Trail Celeron based Soc platform to interface a specific instrument via USB.

To address all these platforms in an effective and straightforward way, ensuring the consistency of the operating system, the device drivers and the application software, a comprehensive approach is desirable.

GNU/LINUX

The usual approach to GNU/Linux consists in the installation of one of the many available distributions. Yet, most of them do not support all the required architectures and instruction sets and, very often, rely on a generic instruction set to achieve improved compatibility at the expense of the performance.

Using distinct distributions for different architectures, which is the current scenario at Elettra, entails a number of possible issues, among which different releases of the system libraries, of the system initialization, e.g System V init rather than upstart or systemd, and of the filesystem hierarchy. Furthermore, diverse bugfix policy is often in place, making the system and platform adimistration quite onerous in the medium/long term.

Sticking to a specific GNU/Linux distribution release, conversely, exposes to the risk of outdated software, especially concerning the kernel, the system libraries and the compiler. Also, recent hardware is usually poorly supported, or even unsupported, by old distributions without a big effort in back-porting.

THE YOCTO PROJECT

The Yocto Project [4] is an open source collaboration, aimed at providing templates, tools and methods to create custom GNU/Linux based systems for embedded hardware, regardless of the architecture. Established in 2010, it involves many hardware manufacturers, open-source operating system vendors and electronics companies. The Yocto Project provides a complete embedded Linux development environment, with tools, metadata and documentation for free, including core system component recipes provided by the OpenEmbedded project. Specific platform support is provided by the Board Support Package (BSP), for which a standard format has been developed. In addition to the very effective command line tools, the Yocto Project provides an Eclipse IDE plugin and the Hob graphical user interface.

FLOP

Starting from the Yocto Project release 1.8, a control system oriented GNU/Linux distribution named Flop has been created. The current Flop release is 0.5, based on the 3.14 Linux kernel, glibc 2.21, and systemd 219; the compiler in use is gcc release 4.9.2. The TANGO Controls framework is also included, featuring OmniORB 4.1.6, ZeroMQ 3.2.4 and TANGO 8.1.2.

Flop Recipes

In the Yocto Project, the key mechanism that allows to define a specific platform support is referenced as recipe. Each recipe specifies the procedures to select and compile software packages. Recipes can be grouped in layers, and multiple layers can be specified to introduce dependencies. Recipes, usually written as shell scripts o Python scripts, make use of common variables that can be accessed by

ISBN 978-3-95450-148-9

different recipes or from configuration files. These features make possible to specify and build systems enabling only very specific functionalities of the required subsystems, whereas the standard distribution package based approach usually enables all the functionalities. Thus, useless, redundant or unwanted features can be easily excluded, improving the system reliability and, as a side effect, decreasing the footprint. As an example, the recipe written for TANGO is listed below.

```
DESCRIPTION = "TANGO is an object oriented distributed control \
  system using CORBA (synchronous and asynchronous \
  communication) and zeromq (event based communication)"
HOMEPAGE = "http://www.tango-controls.org"
LICENSE = "LGPLv3+"
LIC_FILES_CHKSUM = "file://COPYING.LESSER; \
  md5=6a6a8e020838b23406c81b19c1d46df6"

PR = "r0"

DEPENDS += "zlib bash omniorb zeromq"
RDEPENDS_${PN} += "bash"

S = "${WORKDIR}/tango-${PV}"

SRC_URI = "http://download.sourceforge.net/project/tango-cs/
  tango-${PV}c.tar.gz"

SRC_URI[md5sum] = "3dbcc2cf34f8c9395ee72f4ee5ae05dc"
SRC_URI[sha256sum] = "0149e797e5745b1dd8d5d39260889b6da31c8
  4c75c272372255ae8ca3507a116"

inherit autotools pkgconfig systemd

EXTRA_OECONF_append = " --disable-jpegmmx --disable-static \
  --disable-java --without-java --disable-dbserver \
  --disable-dbcreate --enable-stdcxx11=no"
CXXFLAGS_prepend = "-std=gnu++98 "

SYSTEMD_SERVICE_${PN} = "starter.service stopper.service"

do_configure_prepend() {
  ( cd ${S}; ${S}/bootstrap )
}

do_install_append() {
  install -d ${D}${systemd_unitdir}/system
  install -m 0644 ${WORKDIR}/starter.service \
    ${D}${systemd_unitdir}/system
  install -m 0644 ${WORKDIR}/stopper.service \
    ${D}${systemd_unitdir}/system
}
```

In addition to defining some descriptive keywords, such as `DESCRIPTION` or `HOMEPAGE`, the recipe contains the checksums of the specified licence and source files as well as the dependencies toward other tools or libraries. Extra configuration directives for the autotools and the compiler are also specified.

Everything that needs to be included in an embedded system distribution based on the Yocto Project, and therefore in Flop, have to be specified in a recipe. Accordingly, the whole system, built on the basis of the recipes, is univocally defined and characterized by the release number.

Multiple Platform Support

The first advantage of the Yocto Project approach is that Flop support includes all the embedded systems and microprocessor architectures used at Elettra: four generations of PowerPC based VME single board computers manufactured by Artesyn, formerly Emerson/Motorola, the MVME5110, MVME6100, MVME7100 and MVME2500, as well as the ARM based BeagleBone embedded system and the INTEL Bay Tray based Jetway JBC311U93 Soc. Flop provides specific platform support for mathematical coprocessors, vector accelerators, such as the Altivec engine or the SSE4, and optimized intruction sets. The above is especially true for the MVME2500 single board computer P2010 e500 v2 core QorIQ processor, which single and double precision embedded scalar floating point unit is somehow different from the classic floating point unit and uses the integer register file.

The availability of board support packages (BSP) deserves to be analyzed. Each BSP usually supports a very specific kernel release and, even within a single family of boards made by one manufacturer, often applies to a different kernel release for different board models. This makes the development and the management of kernel code, such as device drivers or kernel modules, more complicated. To solve this problem a back/forward porting of the BSP to the current release of Flop has been done, with the request to the community to include it in the mainline. Similarly, the required device drivers have been ported to the current release of Flop. As an example, some changes to the existing Tsi148 PCI-VME bridge device driver, aimed at adding a more transparent and flexible support whenever several different slave VME boards are installed, have been proposed.

The use of different platforms and different distributions sometimes leads to system software and application software fragmentation. The centralized system software management allows to keep the software aligned and up to date for all the platforms. Flop, being based on the Yocto Project, does not make use of a package based approach. On the contrary, all the required components are listed on a recipe, and the whole system compiled from source based on it. This single source approach, although sacrifices the flexibility of the package based software, guarantees the consistency of Flop with respect to all the architectures, as well as the very same patchlevel.

Development Tools

Interesting embedded boards exist which are low resources and low computing power. Moreover old boards are in use at Elettra, such as the MVME5110, which can be considered low performance with respect to modern systems. Native mode development on these platforms, meaning that the tools and the compiler run on the target system, is often quite slow and sometimes limited, or prevented, by the shortage in memory and the inappropriate storage devices. The Yocto Project allows to generate the required cross-compiler tools for the host architecture, typically fast and cheap INTEL-based workstations or servers. Ubuntu 14.04 LTS, running on a modern INTEL-based 64 bit hardware, has been selected as the operating system for the host platform for Flop. The repetitive tasks connected to software development are therefore made on powerful multiuser systems with plenty of memory and fast storage, with an

ISBN 978-3-95450-148-9

explicit benefit for the developer convenience and time savings.

Moreover, some special subsystems may require very specific tools that are often not included in standard GNU/Linux distributions. An example is the programmable real-time unit and industrial communication subsystem (PRU-ICSS) of the BeagleBone. This engine consists of dual 32-bit RISC cores with memory, interrupt controller and internal peripherals. In addition to the supported real-time industrial protocols, such as EtherCAT [5], PROFIBUS and PROFINET [6], EtherNet/IP [7], Ethernet Powerlink [8] and SERCOS III [9], the PRU subsystem has been exploited for some special applications at Elettra Sincrotrone Trieste. These include driving piezo-electric actuated mirrors in real-time for the Fermi seed laser trajectory feedbacks and interfacing the new power supply controllers for the Elettra storage ring [10]. Therefore, the PRU C compiler has been included in the Flop SDK, ensuring the straightforward availability as well as the versioning of the tool with Flop.

On-board Storage

Several partitions are foreseen for the on-board storage; as described hereafter, the current release of Flop requires at least four partitions: one for the boot loader, two for the system image, one for data when in standalone operation without networking.

One of the key features of Flop, as appears from the considerations in the previous sections, is the mandatory versioning which applies to the whole SDK and, of course, to the target platform support. Strict versioning means that systems with the same version must be identical. This aspect can be a limitation whenever a large number of hosts is involved and the application software, even limited to configuration files, is subject to change quite often.

To overcome this restriction the root filesystem of Flop relies on two physical partitions and on the overlay filesystem [11]. The overlayfs allows one, usually read-write, directory tree to be layered onto another read-only directory tree. All modifications go to the upper, writable layer. In Flop, the "lower" directory tree is used for the read-only system image. The "upper", used read-write, stores the changes with respect to the versioned image and is mounted over the previous one exploiting the overlayfs.

Flop also provides a simple and reliable mechanism for system update, with rollback capabilities. Two partitions are foreseen for the Flop system image, one of which is active. A major update can be performed, on a running system, writing the new Flop image to the inactive partition and reconfiguring the system to boot from that one. On next reboot the boot loader will load the updated system. It's worth noting that the specific application and configuration data, stored in the read-write partition by means of the overlayfs, are preserved because not affected by the update procedure. Should it emerge any issue with the updated system, the rollback to the previous image is straightaway and just implies the reset of the boot parameter and the system reboot. The local storage layout and the interactions with the network share, described in the next section, are shown in Fig. 1 for the network operating mode case.

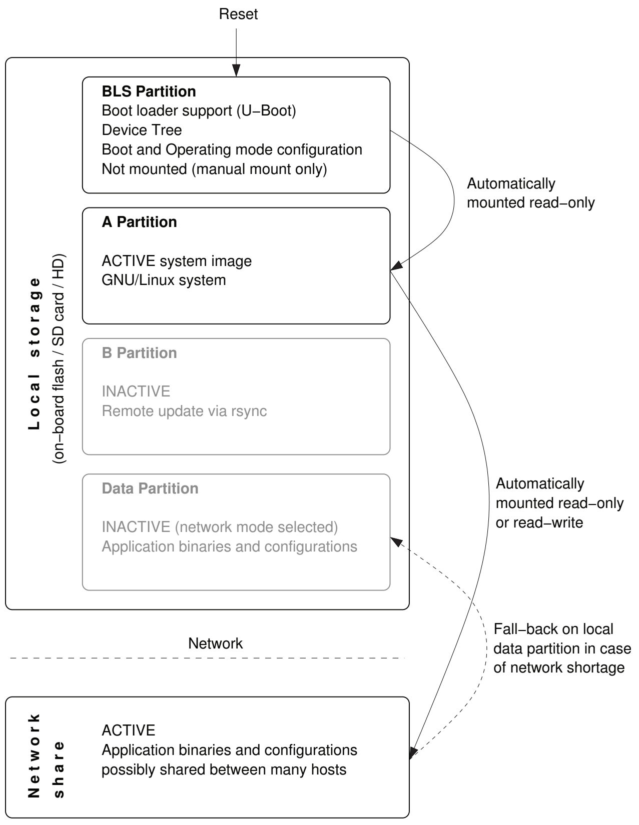
Reset
BLS Partition
Boot loader support (U–Boot)
Device Tree
Boot and Operating mode configuration
Not mounted (manual mount only)
Automatically mounted read–only
A Partition
ACTIVE system image
GNU/Linux system
B Partition
INACTIVE
Remote update via rsync
Data Partition
INACTIVE (network mode selected)
Application binaries and configurations
Automatically mounted read–only or read–write
Local storage
(on–board flash / SD card / HD)
Network
Fall–back on local data partition in case of network shortage
Network share
ACTIVE
Application binaries and configurations possibly shared between many hosts

Figure 1: Local and network storage layout and interactions.

Standalone/Network Operation

Flop has been designed to support both network and network-less, or standalone, operation, where a network enabled setup is not limited to the configuration of the IP address but also includes DHCP, NFS, NIS, NTP, SSH, HTTP and FTP support. Both operating modes have to be addressed in a simple and effective way, minimizing the complexity from the user point of view. Configuration files, binary executables and, possibly, kernel modules, especially when related to specific user applications, can reside in different locations of the filesystem, depending on the selected operating mode. The solution found provides an additional partition to store all the configuration files, required by the boot loader, in order to specify and select the operating mode. If the standalone mode is selected, Flop makes use of a local partition for the data storage; if, conversely, the network mode is selected a part of the filesystem can be shared over a network. Currently, the NFS is not compatible with the overlay mechanism in use; the most suitable network protocol is, at the moment, 9P [12].

When in network mode, essential services, such as IP configutation, DNS, routing and NTP, can be configured via DHCP. In addition, an approach based on DHCP custom options is under evaluation to add the support for a number of parameters, not included in the DHCP specification, that will allow a complete centralized management of the system configurations. Precision Time Protocol (PTP) support is also available and, even not included by default, can be exploited if requested.

Moreover, not excluding the possible adoption of advanced remote management systems such as Puppet or Chef, an HTTP/FTP based interface, aimed at system configuration, is under development. Whenever required, a light http server can be deployed to provide a simple web page exposing the main system configuration parameters and, possibly, any configuration for specific user applications.

CONCLUSION

The increasing number of different embedded systems in use at Elettra, as well as the need of comprehensive system software updates, lead to the need of a unified approach for embedded platform management. Based on the Yocto Project, Flop, a control system oriented GNU/Linux distribution hase been designed. Flop fulfils many of the requirements of a modern embedded operating system, provides a user friendly development environment and rationalizes the support for different platforms and architectures, simplifying the administration and ensuring the consistency.

REFERENCES

[1] D. Bulfone et al., “New front-end computers based on Linux-RTAI and PPC”, ICALEPCS’03, Gyeongiu, Korea (2003).

[2] C. Scafuri, L. Pivetta, “The evolution of the Elettra control system”, ICALEPCS’07, Knoxville, USA (2007).

[3] M. Lonza et al, “The control system of the FERMI@Elettra free electon laser”, ICALEPCS’09, Kobe, Japan (2009).

[4] Yocto Project website: `https://www.yoctoproject.org`

[5] EtherCAT Technology Group website: `https://www.ethercat.org`

[6] PROFIBUS and PROFINET website: `http://www.profibus.com`

[7] EtherNet/IP website: `https://www.odva.org/Home/ODVATECHNOLOGIES/EtherNetIP.aspx`

[8] Ethernet Powerlink website: `http://www.ethernet-powerlink.org`

[9] SERCOS III website: `http://www.sercos.com/technology/sercos3.htm`

[10] S. Cleva, L. Pivetta, P. Sigalotti, “BeagleBone for embedded control system applications”, ICALEPCS’13, San Francisco, USA (2013).

[11] Linux kernel overlay filesystem: `https://www.kernel.org/doc/Documentation/filesystems/overlayfs.txt`

[12] 9P website: `http://9p.cat-v.org`