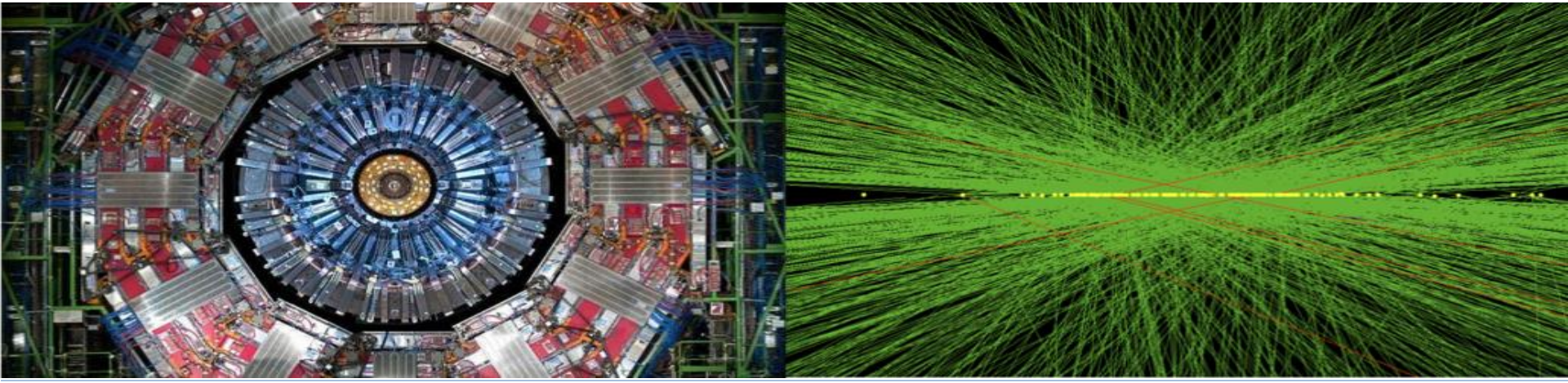
APx
Embedded Linux Developments

J. Tikalsky

June 9, 2021

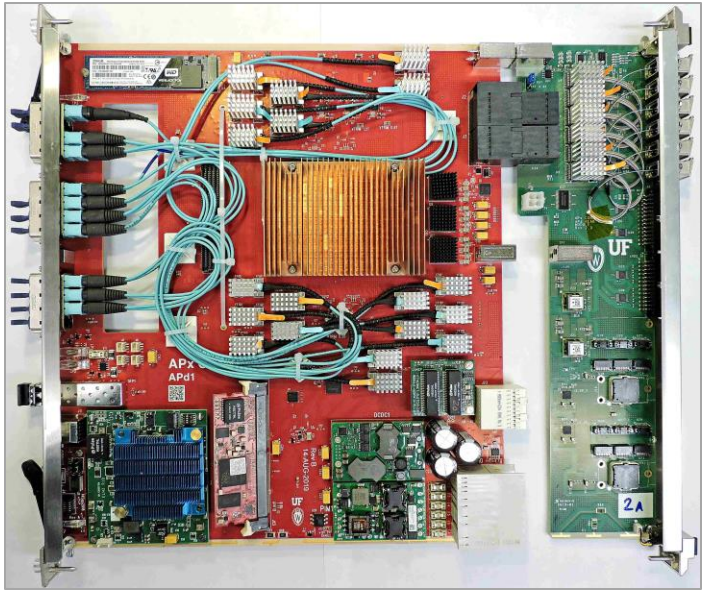


Talk Outline

- APx Hardware Platforms
- CentOS Installation & Configuration
  - Bootstrap
  - Puppet
- Geographic IP Addressing
  - Goals & Purpose
  - Implementation
  - Limitations
- Peripheral Mapping
- Support Requirements

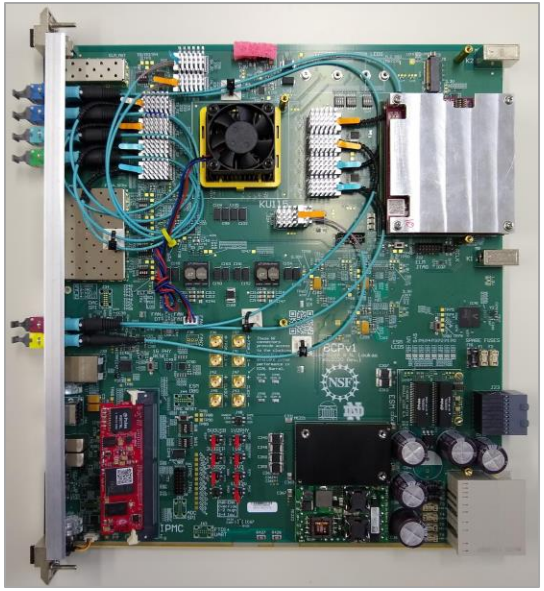
APx Hardware Platforms

CMS Trigger Demonstrator



APx Test Hub

CMS Barrel Calorimeter Demonstrator



APx Embedded Linux Mezzanine (ELM)

ELM1 (XC7Z045 based)

- A series of mezzanines with a single purpose and form factor, which vary in cost and power
  - ELM1: ZYNQ-7000, by U-Wisconsin
  - ELM2: Zynq UltraScale+, by U-Florida
- 84mm × 75mm form factor
- +12V Supply
- SD/QSPI primary/recovery boot options (IPMC-selectable)
- 512MB DDR3 Memory

CentOS: Installation

- *Step 1:* Boot "recovery" image via IPMC request (minimal PetaLinux, from secondary flash media)
- *Step 2:* Install script fetches image components based on a provided manifest file
  - Base Disk Image
  - applyprofile (puppet launcher) config file
  - RPM repository definitions
- All components are protected by a sha256sum-based chain of trust from the manifest, allowing the use of HTTP to simplify server setup in lab environments.

CentOS: Install-Time Configuration

- *Step 3:* A minimal subset of the puppet configuration is applied at stage 1
  - yum repository files
  - BOOT.BIN for PL & kernel image
  - tmpfs mountpoints
- *Step 4:* At first (every) boot, `applyprofile` is run to apply all other required configuration
- `applyprofile` tool wraps `puppet apply`, fetching configuration images over HTTP, verifying GPG signatures, adding "environment selection" facts, etc.

Puppet Apply

- Goals
  - Reliable: Consistent, reproducable configuration states
  - Efficient: Easy lab environment setup, low resource use in operation
  - Stable: ***No unexpected changes breaking the system mid-run***
- `puppet apply`
  - No complex puppet server in lab environments. Just HTTP and tar/gpg
  - No runtime memory or CPU costs once configuration is applied
- Boot-time configuration
  - Ensures correct configuration while conserving operational resources
  - Once a card is booted its state is constant, ensuring stability during data-taking
  - All changes happen during known/planned windows: Only cold startup or manual interventions change the card state

Applying puppet configuration on an ELM

```
[root@c4-s13-linux ~]# applyprofile
Downloading profile...
Downloading signature...
Verifying signature...
gpgv: Signature made Thu 08 Apr 2021 05:00:51 PM UTC
gpgv:                using RSA key 8F83AAD16F321564
gpgv: Good signature from "UWCMS Release Signing Key"
Notice: Compiled catalog for c4-s13-linux.lan in environment puppet_profile in 7.23 seconds
Notice: Applied catalog in 20.06 seconds
```

APx Puppet Config Module

Hiera hierarchy, with board & location layers

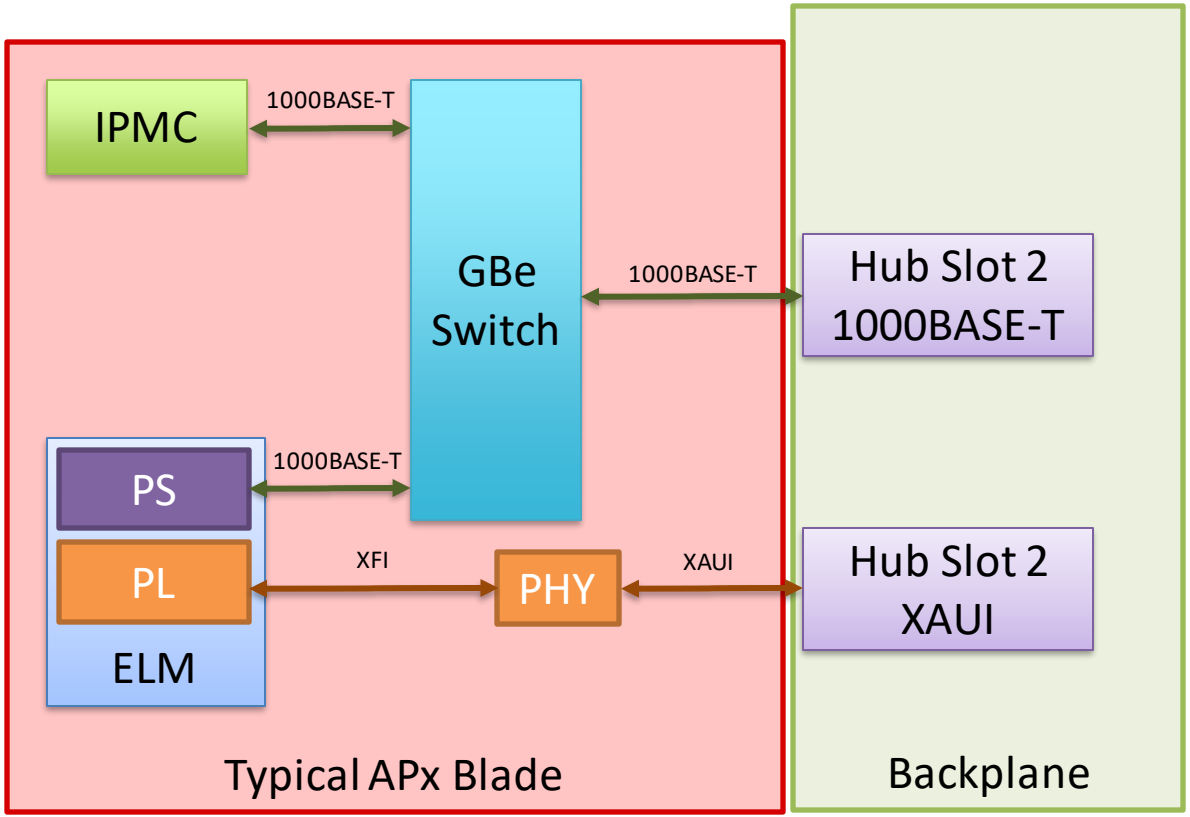
```
hierarchy:
  - name: "Local on-card configuration overrides"
    path: "/etc/puppet-config.yaml"

  - name: 'Location + Mainboard specific
configuration'
    path:
"location_%{facts.apx_location}+mainboard_${facts.apx_
mainboard}.yaml"

  - name: 'Location specific configuration'
    path: "location_%{facts.apx_location}.yaml"

  - name: 'Mainboard specific configuration'
    path: "mainboard_%{facts.apx_mainboard}.yaml"

  - name: "Data for all cards"
    path: "common.yaml"

  - name: "Sensible defaults"
    path: "defaults.yaml"
```

- We produced an APx puppet module which covers base requirements
  - Core basics (NTP, yum repos, requested packages)
  - Users & authentication (kerberos, nfs homedirs)
  - Base APx services (RPC, fpgaloader, sysmap, elmlink, etc)
- Primary configuration through Hiera: puppet mastery not required
- Layers for board-type and location allow easy, flexible configuration

Geographic IP Addressing: Introduction

- What is Automatic Geographic IP Addressing?
  - Names & IP addresses are assigned to locations, not devices
  - Swapping in a spare board is a zero-config process: Software talks to "Crate 1, Slot 5", not "APd1 #14".
  - Previous gen GeoIP is already used by our µTCA boards in operation in CMS.
- More complex than standard MAC Address DHCP
  - Requires a way to identify a board's physical location: IPMI
  - Requires a way to receive an address based on Geographic, rather than Device Identifier (i.e. no MAC address): DHCP with Client Identifiers
  - A way to identify the address of a specific device (non-geographic access) is useful for expert tools.
- Geographic DHCP Client Identifier Example:
  `\x004516GV-0001-13-linux` ↔ Crate Serial# Slot # Sub-component

Example Blade Network Layout



GeoIP: Implementation

- *Step 1:* The Embedded Linux Mezzanine (ELM) requests geographic information from the IPMC via a UART link
- *Step 2:* The IPMC uses various strategies to determine its physical location
  - One strategy implemented based on crate chassis serial number. Additional strategies supported.
- *Step 3:* The ELM generates a Geographic DHCP Client Identifier from this information
- *Step 4:* The ELM requests DHCP configuration using this identifier

GeoIP Configuration Logs

```
Starting Configure Geographic DHCP Client Identifier...
Strategy preference order: ANY, FAIL
Requesting GeoLocation from IPMC.
Available GeoLocation strategies:
  Rank  0, Strategy "first_rackmount_chassis_serial": Crate "4516GV-0001", Slot 13.
  Rank  1, FAIL
Established GeoLocation: Crate "4516GV-0001", Slot 13.
Client identifier "\x004516GV-0001-13-linux" / 0:34:35:31:36:47:56:2d:30:30:30:31:2d:31:33:2d:6c:69:6e:75:78
Updating /etc/dhcp/dhclient.conf
Updated /etc/dhcp/dhclient.conf
```

GeoIP: Challenges – Definition

- Multiple endpoints may require configuration. All need a way to receive GeoLocation information.
  - IPMC, Linux Control Processor, FPGA-based endpoints
- The embedded bare-metal `lwIP` stack (if used by the IPMC) does not support DHCP Client Identifiers (MAC address only)
- Cards must occasionally be accessed by hardware identity, rather than geographic identity
  - In µTCA era, GeoIP did not require DHCP, and assigned a second address, in addition to the DHCP/MAC based default
  - Only one DHCP-based IP address per NIC is easily supported in CentOS

GeoIP: Challenges – Current Plans

- All APx boards are designed such that the embedded Linux system can communicate with all networked components (via UART, AXI Chip2Chip, etc)
- We currently do not plan to use GeoIP on the IPMC
  - Accessing the IPMC is required only as a part of expert maintenance, not standard operation.
  - Each device gets the addressing scheme relevant to its operation.
  - The ZYNQ-IPMC and ELM can request status information from each other via their multi-purpose UART link, including IP address information, which can be used to locate the other as needed.
- We are considering implementing a centralized device identity database where cards can record their identity and address information on startup
  - Maintains the core principles of GeoIP zero-config addressing, while supporting nonstandard modes of access
  - Available for use by expert tools in maintenance and diagnostic operations
  - Relevant *only* for expert maintenance. **Not required for operation.**

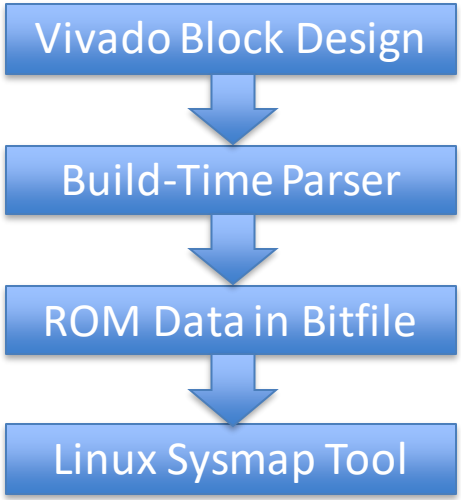
Sysmap: Motiviation

- Modern SoC-based designs present a multitude of varying devices to the processor for integration.
- Labels for these busses (even with device tree configuration) do not automatically cross into Linux.
  - Which bus is i2c-15?
  - Where is the reset GPIO for my Firefly?
  - Where is the UIO segment for the fpgaLoader IP core?
- Device numbers are *not stable between PL builds*
- **Goal: Provide meaningful and consistent device naming and access**

Excerpt of "`ls /dev`" on an APd1

```
/dev/gpiochip0   /dev/gpiochip13  /dev/gpiochip2  /dev/gpiochip7  /dev/i2c-10  /dev/i2c-15  /dev/i2c-2   /dev/i2c-24  /dev/i2c-29  /dev/i2c-7  /dev/uio2  /dev/uio7
/dev/gpiochip1   /dev/gpiochip14  /dev/gpiochip3  /dev/gpiochip8  /dev/i2c-11  /dev/i2c-16  /dev/i2c-20  /dev/i2c-25  /dev/i2c-3   /dev/i2c-8  /dev/uio3  /dev/uio8
/dev/gpiochip10  /dev/gpiochip15  /dev/gpiochip4  /dev/gpiochip9  /dev/i2c-12  /dev/i2c-17  /dev/i2c-21  /dev/i2c-26  /dev/i2c-4   /dev/i2c-9  /dev/uio4
/dev/gpiochip11  /dev/gpiochip16  /dev/gpiochip5  /dev/i2c-0      /dev/i2c-13  /dev/i2c-18  /dev/i2c-22  /dev/i2c-27  /dev/i2c-5   /dev/uio0  /dev/uio5
/dev/gpiochip12  /dev/gpiochip17  /dev/gpiochip6  /dev/i2c-1      /dev/i2c-14  /dev/i2c-19  /dev/i2c-23  /dev/i2c-28  /dev/i2c-6   /dev/uio1  /dev/uio6
```

Sysmap: Architecture



- A BRAM at a fixed address stores device information in the PL.
- At bitstream generation time, the block diagram address tree is automatically analyzed, compressed, and stored in the BRAM.
- At boot-time, a sysmap process reads this data, and intelligently configures Linux devices based on a configuration file
  - Sets up symlinks with meaningful device names
  - Sets device permissions
  - Walks through I2C muxes, etc. defined in the Linux device tree
- There are plans already underway to extend this to processing FPGAs

Excerpt of "`ls -l /dev`" on an APd1

```
lrwxrwxrwx 1 root root           9 Jun 22  2018 gpio.axi_gpio_firefly_rst -> gpiochip5
lrwxrwxrwx 1 root root           5 Jun 22  2018 i2c.axi_iic_bridge.70.01.71.03 -> i2c-19
lrwxrwxrwx 1 root root           4 Jun 22  2018 uio.fpgaLoader_0 -> uio8
```

Device "axi_iic_bridge"
Mux at 70h set to 1
Mux at 71h set to 3
(Muxes defined in Linux Device Tree)

Support Requirements: APx Operation

- Standard Network Services
  - DHCP, NTP (all environments)
  - DNS (production – strongly recommended in lab environments)
- Configuration & Bootstrapping
  - HTTP – Static Content (all environments)
  - All configuration is HTTP based for ease of support and bringup in both lab environments and production.
- Authentication
  - Kerberos (production – optional in lab environments)
- Package Repositories
  - General CentOS 8 repositories (all environments)
  - APx CentOS 8 repository (all environments)
- GeoIP Device Location Database
  - Custom lightweight APx web service (production – optional in lab environments)

Support Requirements: CentOS Builds

- Official CentOS builds are unsuitable for CERN
  - The future of CentOS is as a rolling release distribution (http://bit.ly/CentOS8Stream)
  - CERN installations require long-term stable releases
  - ARM is not supported by the CentOS Core Project, only an AltArch Special Interest Group
- Many groups using CentOS have a similar requirement
  - Long-term stable releases or snapshots are necessary for experimental infrastructure
  - Both 32bit and 64bit ARM builds are needed by CERN users
    - ZYNQ-7000 (32bit armv7hl)
    - ZYNQ UltraScale+ MPSoC (64bit aarch64)
    - Other devices
  - **This leads to a significant risk of duplication of effort**
- ***Centrally supported CentOS builds for armv7hl & aarch64 would benefit many groups***

Backup Slides

The APx Consortium

- Pooling of efforts in ATCA Processor hardware, firmware and software development
- Multiple ATCA processors and mezzanine board types
- Modular design philosophy, emphasis on platform solutions with flexibility and expandability
- Reusable circuit, firmware and software elements

Custom APx SoC Boards

ZYNQ-IPMC
XC7Z014S/XC7Z020 based

- XC7Z014S/XC7Z020 device
- 244-pin DIMM form factor
- +3.3VMP supply
- 109 PL IOs, 16 ADC channels
- 256MB DDR3

Embedded Linux Mezzanine (ELM)
ELM1 (XC7Z045-based)

- ELM1: XC7045 device
- 84mm × 75mm form factor
- +12V Supply
- SD/QSPI primary/recovery boot options (IPMC-selectable)
- PL IOs: 24+ @ 3.3V, 74 @ 1.8V
- MGTs: 8
- 512MB DDR3 Memory