ISAC EPICS ON LINUX: THE MARCH OF THE PENGUINS

J. Richards, R. Nussbaumer, S. Rapaz, and G. Waters,
TRIUMF, Vancouver, B.C., Canada

Abstract

The DC linear accelerators of the ISAC radioactive beam facility at TRIUMF do not impose rigorous timing constraints on the control system. Therefore a real-time operating system is not essential for device control. The ISAC Control System is completing a move to the use of the Linux operating system for hosting all EPICS IOCs. The IOC platforms include GE-Fanuc VME based CPUs for control of most optics and diagnostics, rack mounted servers for supervising PLCs, small desktop PCs for GPIB and RS232 instruments, as well as embedded ARM processors controlling CAN-bus devices that provide a suitcase sized control system. This article focuses on the experience of creating a customized Linux distribution for front-end IOC deployment. Rationale, a roadmap of the process, and efficiency advantages in personnel training and system management realized by using a single OS will be discussed.

INTRODUCTION - HISTORY

The ISAC control system, implemented using the EPICS toolkit, is completing a migration to use only the Linux operating system for both channel access clients and IOCS. This differs significantly from the original system concept of 1997 in which all IOCS were Motorola VME CPUs running VxWorks, SUN/Solaris was used for the development and production servers, and operator console displays ran on Windows PCs acting as X-terminals [1].

To solve reliability problems, Redhat Linux desktop distributions replaced Windows as the operating system on the operator consoles as early as 2003. Also in 2003, the first i486 processors in the PC104 form-factor and running VxWorks were deployed as IOCs that interfaced with PLCs using TCP/IP [2]. VxWorks boot parameters were installed into the PC104 BIOS via the JTAG interface, which required legacy parallel port hardware and Windows software. To replace or re-function a unit required a new JTAG download via a dongle. By 2005, more powerful Pentium III-based VME CPUs (GMS Mariner – 64 MB RAM) running VxWorks were introduced as replacements for selected MV162 (16 MB RAM) installations.

At that time it became apparent that the maintenance and knowledge support for three different operating systems (Windows,Solaris,VxWorks) was putting too much strain on the resources of the small controls group. As EPICS provided operating system independence for IOCs with release 3.14, a long-term goal was formulated to use Linux as the only operating system in the ISAC control system. Linux is feasible because ISAC is essentially a DC machine which does not demand hard real-time performance from the control system.

Experience using Linux as the operating system on servers, consoles, and developer desktops proved that this single operating system could meet most control needs. Over the period 2004-2010 all the SUN/Solaris hardware was retired and replaced with rack mounted servers running Scientific Linux. Virtualization technology on servers allowed services to be split into single-function servers (database, web, file, dns, ldap) with precise access control. Linux also reduced annual licensing and hardware costs.

The final frontier for Linux deployment was the IOC. Problems with instability on the PC104's VxWorks 5.4 network stack led to replacement with rack-mounted servers running Scientific Linux (SL) which host multiple IOCs. This architecture/OS model met the IOC criteria for stability, reliability, and extensibility. Deployment of single-purpose IOCs on PCs running Redhat desktop distributions began in 2005 [3]. These Linux PCs were used to both develop the IOC application and to deploy it in the field. This "unclean" combination of development and production machine led to the necessity of having "hot spares" available. The PC towers were also prone to hardware failures of hard drives, fans, as well as other problems encountered in the sub-optimal accelerator-laboratory floor environment. As the number of these PC installations grew, so did the effort of Linux administration.

A PROTOTYPE IOC RUNNING LINUX

The need for a small form-factor, low power, no-moving-parts IOC host was identified, and the TS-7350 [*]ARM-based Single Board Computer (PC104 form factor) was chosen as an evaluation platform. Methods for performing software development and field deployment were developed, tested, and deemed acceptable

A project that required a small and highly portable control system for 4 power supplies and 2 vacuum gauges required a microprocessor that was robust, with no fans or spinning media. The ARM units met these criteria. Technologic Systems provide a development kit with a customized Debian Linux distribution and proprietary drivers. A daughter board, TS-CAN1, enabled connectivity to a CAN-bus network using the open-source Ocera-lincan[†] driver, and in-house EPICS device support modelled after existing CAN-bus support. The result was a very small system, with the ARM IOC booting from SD flash within 40 seconds, connected to 5 CAN-bus-attached microcontrollers. A laptop hosting EPICS

[*] http://www.embeddedarm.com/products/board-detail.php?product=TS-7350
[†] http://www.ocera.org/download/components/WP7/lincan-0.3.3.html

channel access extensions serves as an operator console and boot host. This "Control System in a Suitcase" could be deployed for acceptance tests on factory floors. It provided the background experience for identifying the way we want all fieldbus-connected Linux IOCs to work.

EMBEDDED X86 IOC REQUIREMENTS

From the experience gained with the installations mentioned above the following requirements for a generic embedded Linux-x86 IOC were defined (rationale in parentheses):

- No moving parts (increase reliability)
- Network boot – no dongles (simplify maintenance)
    - Reboot time < 40 s (derived from ARM)
- Application agnostic hardware. The target computer does not associate with an application until booted. (eliminates a dedicated "hot spare" per IOC)
- A Linux distribution tailored to ISAC Controls IOC needs (i.e. **Isaclinux**).
    - known kernel composition with all functionality understood
    - kernel to include only services required for our IOC objectives
    - must use standard kernel source distributions
    - root file system must be minimal in size, exposing minimal utilities (reduce security risks, fast boot time)
- Replace the functionality of currently installed VxWorks IOCs
- Separation between application development and production environment – analogous to the VxWorks model where products are created by a cross-compiler on a development host and deployed to different target at run time

METHODS

- The hardware selection for the VME bus was VMIC 7648(Pentium III, 1260 MHz, 512K cache, 128 MB RAM) which we acquired at discount pricing. For RS232/RS485 and GPIB buses we selected a mini ITX motherboard (Atom N270, 1.6 GHz, 512 K cache, 1 GB RAM). These units have low power consumption, low heat load and solid state drives which should translate to high reliability. The ITX provides 2 exposed serial ports, plus an addition 4 on the motherboard. It also provides 2 PCI slots for expansion cards for such buses as GPIB and CAN. This product was considered to have a reasonably long product future and hence a high availability of supply.

- Both hardware architectures selected allow for network boot via their BIOS setups. The industry standard Preboot Execution Environment (PXE) protocol is implemented with pxelinux[‡] and TFTP for diskless, network-only, booting, The VMIC boot time averages 36 seconds. The ITX/Atom pxelinux boot is much slower, 140 seconds. However, by using the enhanced utility, gpxelinux, the ITX/Atom boots in approximately 40.
- Agnosticism was achieved by using a centralized boot system and common root file system. Kernel command line arguments are exposed to the /proc file system and can be accessed by user space applications. A single user space script file "/etc/init.d/rcS" associates kernel command line keywords with functionality. This script is easy to customize, easy to understand, and well structured. Custom keywords are used to identify scripts to run prior to launching an IOC application, such as GPIB or Ethernet private network setups. Other keywords identify the IOC application boot directory and start-up script.
- Customization of an existing Scientific Linux distribution was attempted but abandoned when the difficulties of obtaining kernel sources was apparent. Rather than paring down a bloated desktop release, a very compact Linux distribution was selected as a starting point. Micro Core Linux[§] is a 6MB text-mode distribution of Tiny Core Linux, boots very quickly, uses a Linux 2.6 kernel, and utilizes Busybox[**] for core utilities. The entire Micro Core Linux file system was converted, in-house, to an initrd format. On-going customization of the initrd as well as tools and methods for testing and deployment have rendered the existing incarnation fully disconnected from its ancestry.
  The Micro Core kernel was further pared from 2.2 to 1.8 MB and rebranded as the Isaclinux kernel. Root file system customizations have added 4 MB to the Micro Core starting size, resulting in an Isaclinux file system of 10 MB. The main customization was to the rcS script for launching applications. Other additions for Isaclinux include: ssh server (Dropbear), GNU screen (used to provide a sharable IOC console), C-Kermit (useful troubleshooting tool for serial I/O), EPICS libraries and utilities (e.g. camonitor, caRepeater), and dependency libraries (e.g. libreadline, libncurses, libpcre).
- EPICS OSI library routines for VME are only available for VxWorks and RTEMS. Thus, a

[‡] http://www.syslinux.org
[§] http://distro.ibiblio.org/tinycorelinux
[**] http://busybox.net/

port of a number of EPICS drivers from VxWorks to Linux was undertaken. For the VMIC7648 IOC target, the GE-Fanuc VME_UNIVERSE kernel driver is available[††]. Using the driver's API library a support library for EPICS applications was created that includes a set of access routines that create access windows to the VME bus and convert bus addresses to local addresses for specific VME devices (e.g. sysMapVmeMaster and sysBusToLocalAdrs), interrupt connect routines, and OSI wrappers for VME access. The use of OSI wrappers minimized the software changes required when porting VxWorks device drivers to Linux. Since the VME bus is a Big-endian bus and x86 CPUs such as the VMIC7648 are Little-endian, it was necessary to add byte swapping macros to all VME access operations.

A number of VME device support software changes were also required. For the CAN-bus TEWS Tip810 driver connected via TVME200 VME-bus industry pack carriers, the message receive task was throttled to prevent CPU hogging. The OMS58 receives commands via a 256 byte circular input buffer. On the Pentium based VMIC7648, command acceptance time-outs made it necessary to throttle the send_mess() function in the OMS58 driver code using the POSIX standard function nanoSleep().

- Separation of development and production environments is clearly met by the Isaclinux deployment model. The system is designed to run from a RAM copy created at boot time. Besides being fast, this protects system files from changes and ensures a pristine system on every reboot. This is analogous to the VxWorks model where the VxWorks OS is loaded over the network at boot time. Isaclinux changes can only be introduced via the initrd file system produced on the development server. A Debian development host having the same kernel vintage (2.6.33) as Isaclinux is used to produce the EPICS applications. Using the same C Library and compiler ensures compatibility between run-time environment and EPICS applications

RESULTS – LESSONS LEARNED

One of the primary goals for the Isaclinux project was to standardize deployed systems and stop the proliferation of different hardware architectures, operating system distributions and versions, thereby minimizing system administration overhead. This has been achieved. The development process also offered an opportunity for all authors to become familiar with the internals of the Linux operating system.

The need to customize kernels led to the conviction that a Debian distribution provides a better development platform for embedded systems than Redhat type distributions since access to kernel sources is more standard. The Debian project has official support for embedded systems, including support for the ARM microprocessor. Debian distributions include more up-to-date software related to software development (compilers, libraries, etc.). On the rack-mounted IOC servers that require only TCP/IP to connect to PLCsScientific Linux is still used as the operating system.

We still have some dependency between what we build and where we build it – that is, we have not achieved a cross-native compiling. It would be nice to have a toolchain that creates the target kernel, libraries and apps that could be deployed on any Linux server but our project time ran out. Further enhancements that we are developing include a deployment infrastructure. This consists of packaging the root file system in initrd, version control (legacy/stable/test) of each kernel and root file system, benchmark and regression testing on development hardware and a standardized deployment procedure for production. A configuration tool that provides for event logging of changes to dhcp, pxe, kernel, and initrd is under development.

The VMIC deployments have some issues with certain hardware models. Joerger VSC16 scalers occasionally generate "uninitialized interrupt messages". VxWorks handles these gracefully and performance is not affected. However on Linux these result in the scaler locking on vector 255. A non-elegant stop gap measure that limits us to one scaler only in any VME crate is to always use vector 255. The CAN-bus interrupt handler works well on 1260 MHz VMICs but blocks erratically on 928.1 MHz clock models.

To date we have Isaclinux deployed in a single VME crate and are awaiting a shutdown period to allow replacement of more VxWorks installations. A number of ITX/Atoms that use StreamDevice for GPIB, RS485, and Ethernet fieldbuses have been installed in the control system. We are evaluating their suitability as digital power supply controllers. For the Atoms, we have also implemented Wake-On-LAN to allow remote restart.

The utility of the ARM with its smaller form-factor is still being explored. Although it is not network-boot enabled (uses boot flash), it loads its applications from an NFS mount. We also have found it to serve well as a serial console for other IOCs, and as a CAN-bus loop packet sniffer.

CONCLUSIONS

The ISAC Control System has a response rate requirement of only 10 Hz. This can be met by an operating system such as GNU/Linux that is not classified as real-time. To minimize IOC application pre-emptions, it is advisable to have the IOC Linux host run only essential services. Isaclinux is a small footprint,

[††] Customizations by TRIUMF DAQ Group.

customized version of a 2.6 Linux distribution. It provides minimal services such as SSH, NTP, DHCP, and NFS needed for a network booted IOC. Its use on such platforms as the GE-Fanuc and Atom allow control of devices on fieldbuses such as VME, GPIB, RS232, and Ethernet.

Linux is now used in all tiers of the ISAC Controls task model including servers, operator consoles, and IOCs. One operating system, understood by all controls personnel, allows a small group of control engineers to share maintenance and development duties easily. This strategy facilitates a sharing of administration duties, concentrates on one platform only for device drivers, and has already proven invaluable in troubleshooting situations.

REFERENCES

[1] R. Keitel, M. Leross and G. Waters, “Control system prototype for the ISAC radioactive beam facility,” ICALEPCS97, Beijing, 1997

[2] G. Waters and R. Nussbaumer, “TRIUMF/ISAC EPICS iocs using PC104 platform,” ICALEPCS03, Gyeongjiu, 2003

[3] R. Nussbaumer and G. Waters. “BACnet support for EPICS,” ICALEPCS05, Geneva, 2005