ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA078

A VIRTUALIZED BEAMLINE CONTROL AND DAQ ENVIRONMENT AT PAL

S. W. Kim*, H. J. Choi, H. S. Kim, and W. W. Lee†
Pohang Accelerator Laboratory, Pohang, South Korea

Abstract

At least three different computers are used in the beamlines of PAL, first for EPICS IOC, second for device control and data acquisition(DAQ), and third for analyzing data for users. In the meantime, stable beamline control was possible by maintaining the policy of separating applications listed above from the hardware layer. As data volumes grow and the resulting data throughput increases, demands for replacement of highly efficient computers has increased. Advances in virtualization technology and robust computer performance have enabled a policy shift from hardware-level isolation to software-level isolation without replacing all the computers. DAQ and analysis software using the Bluesky Data Collection Framework have been implemented on this virtualized OS. In this presentation, we introduce the DAQ system implemented by this virtualization method.

VIRTUALIZATION

As the paradigm shifts from single-core to multi-core programming, interest in virtualization is increasing. The beamline control software were categorized into three major parts, IOC / DAQ / Analysis, and has been run on different computers. However, as multi-core processors become common, the traditional approach is inefficient in terms of not using idle computing resources. By dividing the system resources through virtualization, it can allocate dynamically to the environment with high resource utilization and save time for installation and programming environment construction. Unlike server virtualization, a workstation requires terminals such as monitors, keyboard, and mouse to control each virtual machine. Allocating physical hardware to a virtual machine is called *passthrough*. The types of hypervisors will be discussed in the following session.

Virtualization Solutions

Virtualization solutions can be classified into three major types depending on the hypervisors: KVM, Xen, and ESXi (See Table 1). Kernel based Virtual Machine (KVM) converts the Linux kernel into a hypervisor, and takes advantage of Linux's process and memory management, network, I/O, and driver features. KVM has been integrated into the kernel since Linux version 2.6.20 and can be used immediately without rebuilding the kernel. Xen, like KVM, turns the Linux kernel into a hypervisor and can also take advantage of the Linux kernel, but requires a new build of the kernel with the Xen module loaded. KVM and Xen are both open-source and available for free on most Linux distributions and BSDs. Libvirt can manage virtual machines running on top of these hypervisors, and can be easily managed with Virt-manager, a GUI interface that utilizes it. ESXi is a hypervisor provided by VMware. The Free version has some limitations, including up to two physical CPUs, a maximum of eight vCPUs per virtual machine (VM), and other API limitations. The hypervisors listed above enable operating system virtualization and passthrough for free.

Table 1: Virtualization Solutions for GPU Passthrough

Solutions	Hypervisor	Pricing
Linux/kvm	KVM	Free
PROXMOX	KVM, LXC	Free
Linux/Xen	Xen	Free
XCP-ng	Xen	Free
Citrix XenServer	Xen	$763.00 / CPU
VMware	ESXi	Free (w/ limit)

Based on these hypervisors, there are commercial solutions that provide virtual machine, network and storage management tools, and commercial support. Some examples are VMware vSphere (ESXi), Citrix XenServer (Xen), XCP-ng (Xen), and PROXMOX (KVM). Citrix XenServer has removed the GPU passthrough feature from the community edition since version 7.3, but it is still available for free in its open source version, XCP-ng. XCP-ng and PROXMOX are free to use all features without any limitations, but security updates and support are available with commercial annual subscription.

Virtual machine orchestration is required to remotely manage and monitor hypervisors installed in various beamlines without having to log in directly. For most cases, it is sufficient to connect to the hypervisor with SSH and manage the VMs via Libvirt and Virt-manager. A virtualized workstation using CentOS 7 / KVM as a hypervisor will be introduced in the following section. For reference, Orchestration solutions include oVirt (KVM), XOA (Xen), and vCenter Server (ESXi).

Virtual Machines on a Workstation

The workstation's resources consisted of 32 logical CPUs, 32 GB of RAM, two graphics card, a PCIe to USB hub, and hard disks have been allocated to the virtual machines. The procedures of setting up CentOS 7 as a hypervisor and GPU passthrough are summarized separately [1]. CentOS 6 (IOC VM), CentOS 7 (DAQ VM), and Windows 7 (Analysis VM) operating systems were installed, respectively, and eight vCPUs and 8 GB of RAM were allocated. And the remaining resources were assigned to the hypervisor. Since the IOC

* physwkim@postech.ac.kr
† lww@postech.ac.kr

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA078

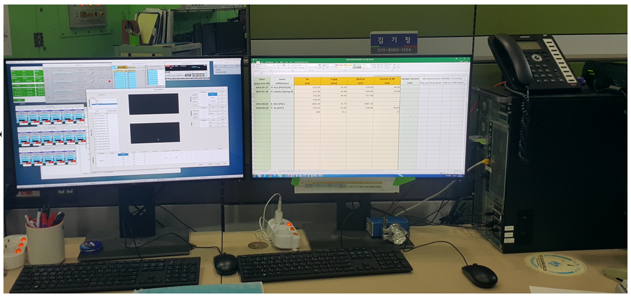
Figure 1: Multiple operating systems are running on one workstation. Headless: IOC VM (CentOS 6), Left: DAQ VM (CentOS 7), Middle: Analysis VM (Windows 7), Right: Workstation.

VM does not require a terminal, it operates in headless mode and allocates Quadro P400 (Primary) and Radeon Wx2100 (Secondary) GPUs to the DAQ VM and Analysis VM, respectively. With PCIe to usb card assigned to the virtual machine, USB devices can be easily added or removed. Figure 1 shows the DAQ VM and Analysis VM running on a virtualized workstation.

DATA COLLECTION FRAMEWORK

Commercial software development tools, such as Labview, Matlab, IDL etc, provide qualified functions, graph libraries, and GUI toolkits as an integrated development environment. Because of these advantages in terms of efficiency and productivity in scientific programming, DAQ and analysis programs have been developed widely using those environments. However since the development environment was not unified, inefficiencies such as code recycling were greatly increased. When a developed scan algorithm is applied to other beamlines, it has to be completely rewritten for the specific environment. For this reason, the Bluesky [2–5] was chosen as the data collection framework for integrating different DAQ environments.

The Bluesky Data Collection Framework

The benefits of choosing a bluesky framework can be listed as follows [2]:

- Ophyd [4] abstracts different hardwares so that these devices can be used in the same way.
- Most of the motors and detectors used in the beamline are included.
- Common scan algorithms using abstracted hardware are predefined.
- Databroker [5] support various backends for storing data and rich metadata.
- Scientific libraries developed in python such as numpy and scipy can easily be used within the bluesky framework.

By using bluesky, common hardware / scans are predefined so that efficiency can be greatly improved by only considering additional features or customizations. In addition, rich metadata can be stored and utilized, which greatly helps program development and user data analysis.

EXAFS with the Bluesky Framework

The beamline 1D at PLS-II has been dedicated to XRD (Hard X-ray Diffraction) and EXAFS (Extended X-ray Absorption Fine Structure) experiment and successfully supported the user community with commercial tools, SPEC (XRD) and Labview (EXAFS). The EXAFS measurement software, previously developed in Labview, was rewritten with utilizing the bluesky data collection framework. The EXAFS program was created with reference to the structure of DAQ software of ISS Beamline at NSLS-II [6].

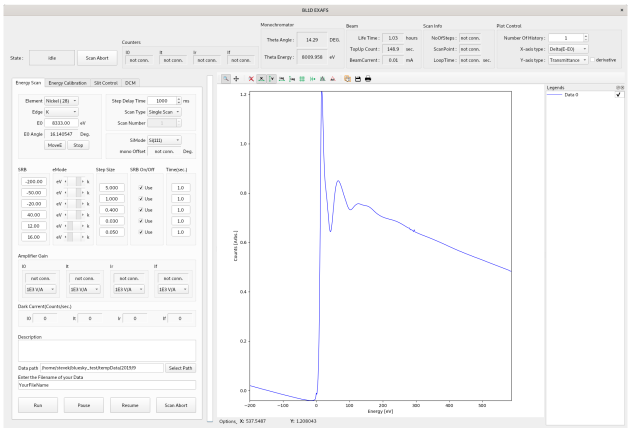

Figure 2: EXAFS data-acquisition software built with the Bluesky Framework [2–6] and the silx [7] plotting library.

The program consists of a three threads, the main thread, the plot thread, and the asyncio thread of the bluesky. The plot thread periodically receives data from the databroker, performs the necessary operations, and draw them on the graph. The bluesky's RunEngine runs on a asyncio thread, performs predefined scan procedures (plan), and sends data to the databroker. The backend of the databroker is mongodb. Normally it works fine with sqlite backend, but mongodb seems to be a better choice if there is a lot of data throughput, such as fly scan.

The silx [7] library was used for the data plot. PyMca's [8] enormous graph and GUI-related codes have been separated into the silx library for use in beamline data analysis programs. The PyMca program currently also uses the silx library. The 1D, 2D, 3D and HDF5 data viewers are provided so silx can be very useful for writing DAQ and analysis programs. In addition, the silx's submitToQtMainThread function can be used to make the GUI more responsive even for heavy loads such as graph updates.

Figure 2 shows the EXAFS measurement program. It is divided into status widget on the top, control widget on the left, and plot widget on the right. The status widget controls the type of x, y axes and the number of graphs in the plot widget. The control widget sets the experiment parameters and passes those parameters to RunEngine, and the scan (plan) is performed on the asyncio thread.

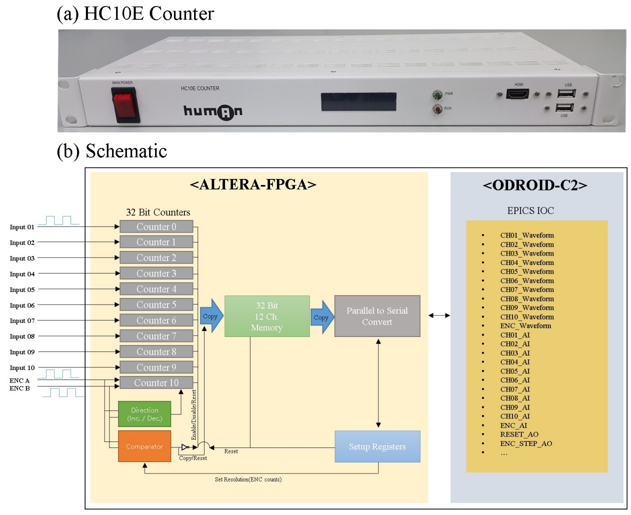

Figure 3: (a): Appearance of HC10E, (b): Internal design schematic.

Fly Scan with FPGA Based Scaler

A fly scan is essential to get the encoder's position and the counts coming from the detectors at the same time. To achieve this goal, a fly scan counter (see Fig. 3) with FPGA and ODROID was designed. The direction of the motor is confirmed by two encoder signals having a 90 degree phase difference, and the operation (Inc. / Dec.) of the position counter is managed. When the position counter reaches the preset value, the detector's counts is copied into the memory. At the end of the transfer, the measurement restarts and the data in memory is transferred to the EPICS IOC via serial communication. The data is stored in the waveform records allocated for each channel.

During the fly-scan, Bluesky's RunEngine saves the newly updated values into the databroker's 'monitor' tables. The plot thread periodically reads data from these 'monitor' tables and displays it on the graph. After the measurement, RunEngine finally reads the waveforms from the scaler and stores it in the 'primary' table. If the 'primary' table exists, the plot thread uses this data instead of the 'monitor' tables. In this way, data could be monitored during measurement. Some monitored data could be lost depending on network traffic and thread conditions. However, the data finally provided to the user is free from this problem because it is read at once from the waveforms of the scaler.

CONCLUSION

Virtualization is an interesting technology in terms of effectively utilizing the resources of multicore processor environment and facillating the construction and operation of a multi-OS in a single workstation. The types of hypervisors and virtualization solutions were discussed, and KVM was used to convert the linux kernel into a hypervisor to build various types of OS on it.

EXAFS measurement software written in labview was rebuilt using the Bluesky data collection framework to unify the fragmented development environment. This confirmed the efficiency and versatility of the virtualization and the Bluesky framework. Therefore, these techniques will be applied to the multiple beamlines at PAL.

REFERENCES

[1] GPU passthrough procedure with KVM on CentOS 7, `https://bitbucket.org/physwkim/kvm_test/src/master/`

[2] *NSLS-II Software Documentation*, `https://nsls-ii.github.io/`

[3] *bluesky*, a Python data collection interface for experimental science, `https://github.com/bluesky/bluesky`

[4] *Ophyd* - EPICS Client Abstractions, `https://github.com/bluesky/ophyd`

[5] *databroker*, unified interface to the various data sources at NSLS-II, `https://github.com/bluesky/databroker`

[6] B. V. Luvizotto, K. Attenkofer, H. Bassan, and E. Stavitski, "XLive: Data Acquisition and Visualization at the NSLS-II ISS Beamline", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, pp. 962–965. `doi:10.18429/JACoW-ICALEPCS2017-TUPHA211`

[7] Thomas Vincent, Valentin Valls, payno, Jerome Kieffer, V. Armando Solé, Pierre Paleo, dnaudet, Guillaume Communie, Pierre Knobel, Marius Retegan, Mauro Rovezzi, Pepijn Kenter, Hans Fangohr, UUSim, Vincent Favre-Nicolin, picca, Captain-Nemoz, woutdenolf, schooft, Tiago Coutinho, Jan Kotanski, Christopher J. Wright, and aicampbell, "silx-kit/silx: v0.11.0: 03/07/2019". Zenodo, 03-Jul-2019. `doi:10.5281/zenodo.3266814`

[8] V.A. Sole, E. Papillon, M. Cotte, Ph. Walter, J. Susini, "A multiplatform code for the analysis of energy-dispersive X-ray fluorescence spectra", *Spectrochim. Acta Part B*, vol. 62, 2007 pp. 63-68. `doi:10.1016/j.sab.2006.12.002`