CROSS-PLATFORM AND CLOUD-BASED ACCESS TO MULTIPLE PARTICLE ACCELERATOR CODES VIA APPLICATION CONTAINERS*

D.L. Bruhwiler#, R. Nagler, S.D. Webb, RadiaSoft, Boulder, CO 80304, USA
G. Andonian, M.A. Harrison, S. Seung, RadiaBeam Technologies, Santa Monica, CA 90404, USA
T. Shaftan, BNL, Upton, NY 11973, USA
P. Moeller, Bivio Software, Boulder, CO 80303, USA

Abstract

Particle accelerator and radiation modeling codes focus on specific problems, rely on complicated command-line interfaces, are sometimes limited to a small number of computing platforms, and can be difficult to install. There is also a growing need to use two or more codes together for end-to-end design or for complicated sub-systems. RadTrack [1,2] is a lightweight cross-platform GUI for such codes, based on the Qt framework [3] and PyQt [4] bindings for Python. RadTrack is designed to support multiple codes, placing no burden on the corresponding development teams. Elegant [5] and the Synchrotron Radiation Workshop (SRW) [6-9] are supported now in a pre-beta stage, and support for GENESIS 1.3 [10,11] is under development. These codes are being containerized via the open source Docker platform [12] for use in the cloud. The open source Vagrant [13] and Virtual Box [14] are used for MacOS and Windows. We discuss RadTrack and our vision for cloud computing.

RADTRACK USER TESTING

In preparation for an upcoming beta test program, user testing of RadTrack was conducted at the IPAC conference in Richmond, VA on May 4 and 5, 2015. In order to ensure the entire team has access to user feedback, we are using Screenflick [15] on the Mac and BB Flashback [16] on Windows to record user actions, audio and video. These videos have been invaluable in discovering basic user interaction (UX) issues. We are now improving the workflow to be more intuitive for end-users.

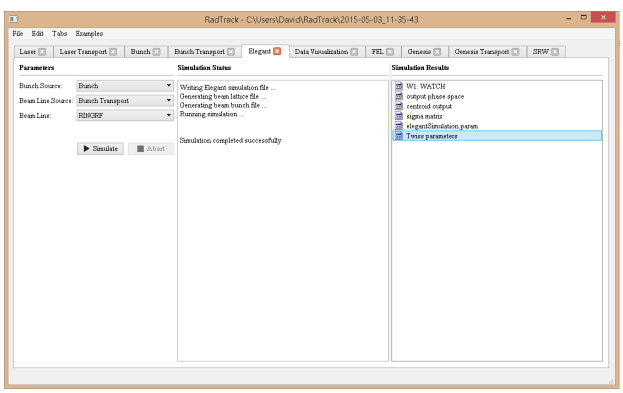

Figure 1: RadTrack Qt widget for Elegant simulations.

The use case presented to test subjects was to build a FODO accelerator lattice from scratch, beginning with a 65 MeV electron beam, and to simulate the problem with Elegant. Figure 1 shows the widget for Elegant modeling. This Tab manages the selection of an accelerator lattice and a beam description, shows runtime output, then provides easy access to the resulting output files for visualization.

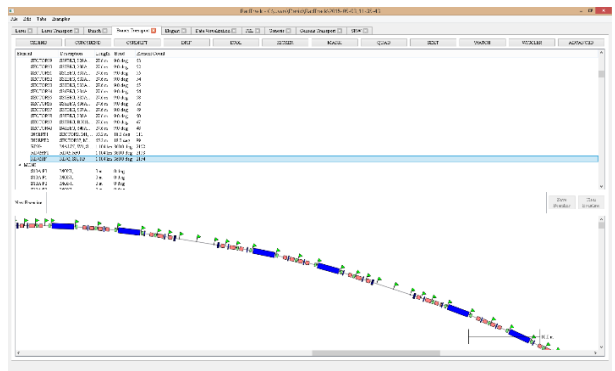

Figure 2: RadTrack Qt widget for drag-and-drop construction and visualization of particle accelerator lattices.

Figure 2 shows the bunch transport tab, which parses Elegant lattice files to present a graphical representation of the beamline. Drag-and-drop features enable interactive modification of existing lattices or rapid creation of new beamlines. The Elegant tab can use the bunch transport tab directly, or parse a specified lattice file. The user can create multiple bunch transport tabs.

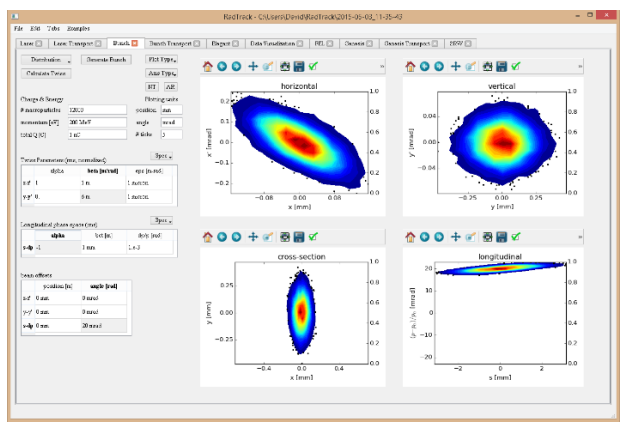

Figure 3: RadTrack Qt widget for interactive specification and visualization of particle beams.

*Work funded by DOE Basic Energy Sciences grant DE-SC0006284.
bruhwiler@radiasoft.net

Figure 3 shows the bunch tab, which parses Elegant SDDS [17] particle files, enabling interactive visualization and rapid creation of new bunch descriptions. The Elegant tab can use the bunch tab directly, or parse a specified particle data file. The user can create multiple bunch tabs.

BEAMS AND RADIATION

An important goal of RadTrack is to simplify workflows involving both electron beams and radiation. For example, a free electron laser (FEL) design tab is provided, as seen in Figure 4. This tab provides estimates of performance, for specified electron beam and undulator parameters. The calculation is based on the universal scaling function of Xie [18], which is a polynomial with 19 fitting parameters.

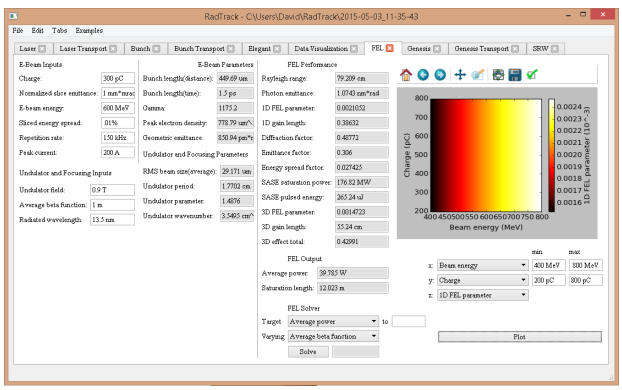

Figure 4: RadTrack Qt widget to explore FEL parameters.

The tab for GENESIS simulations, which is now under development, will use these parameters to model the FEL. We will also enable communication between the bunch tab and the FEL tab, so that the output from Elegant or other tracking codes can be immediately used for FEL design and simulation.

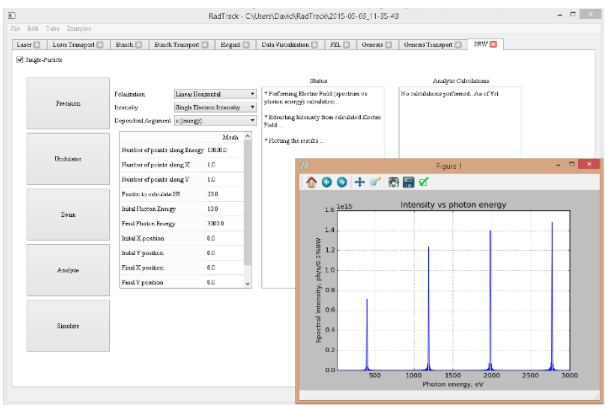

Figure 5: RadTrack Qt widget for synchrotron radiation simulations using SRW.

The SRW tab, seen in Figure 5, is ready for user testing. It enables rapid modeling of fully coherent radiation ('thin' or 'single-particle' mode), as well as the time-consuming treatment of partially coherent radiation ('thick' or 'many-particle' mode). Analogous to the FEL tab, analytic and semi-analytic calculations are presented in the panel on the right, so users can better understand the numerical results.

Although not shown here, RadTrack also includes a tab for laser beams, assuming a Gauss-Hermite expansion [19,20] that satisfies the paraxial approximation [21,22]. There is also a tab for optical transport, closely analogous to the bunch transport tab seen in Figure 2, which will in the future be used to directly support the SRW tab. We also have an optical transport tab to support the GENESIS tab, which can simulate multiple undulator segments and overlaid quadrupoles, as is necessary for X-ray FELs. In the next two years, we will add support for Synergia [23], WARP [24-27], SHADOW [28,29] and other codes.

SCIENTIFIC CLOUD COMPUTING

Our vision is to bring scientific cloud computing to the accelerator technology and radiation source communities. We believe scientific software must be open source, so RadTrack [2] and all of our cloud computing infrastructure will be available on Github and will be developed openly via the Github issues feature [30]. Our vision has three primary components: a) containerized computing – put your simulation in a box; b) the browser is the UI – never install software again; and c) seamless legacy – you won't realize you're in the cloud.

Containerized Computing

Some scientific software development teams have expended significant effort to achieve cross-platform execution on Linux, MacOS and Windows. Examples include SRW, Elegant and GENESIS. However, this is time consuming, expensive, and still requires sophistication on the part of the user to correctly install and use such codes. Dependencies often include a specific version of Python, of Qt, of Python-Qt wrappers and other libraries, which significantly complicate installation and can sometimes clash on any OS with previously installed software.

Because most scientific codes are primarily developed and used under Linux, Windows is obviously the greatest challenge for cross-platform success. However, the MacOS is sufficiently different that it cannot be supported without significant effort on the part of the development team, which often has limited funding and scientific priorities. Even different flavors of Linux can cause serious pain for users trying to install a scientific code, especially when using a cluster where many of the required dependencies are not installed, or the system installation is the wrong release.

Using Docker [12] on Linux, it is possible to create a file that contains a scientific code or codes, plus all required tools and dependencies, which can then be copied to any Linux server or cluster and rapidly activated. A user can ssh into the container, if necessary, or the software can be accessed remotely through a web-based UI. This removes the pain of software installation on Linux, and it enables cloud-based scientific computing by making such codes available on demand, whether it is

ISBN 978-3-95450-168-7

on the local cluster, a supercomputer or a commercial cloud service.

There is no runtime overhead associated with the use of Dockerized software on Linux. On Windows and MacOS, the containerized software must run inside a Linux virtual machine. The open source Vagrant [13] and VirtualBox [14] make this possible with a minimum of runtime overhead, because the robust, cross-platform VM runs in 'headless' mode. We are developing open source single-click installers for RadTrack, to hide these complications from the user.

Figure 6: The concept of containerized computing.

The schematic in Figure 6 attempts to convey the spirit of containerized computing, which enables a number of other compelling possibilities. For example, scientists will be able to archive their entire simulation environment in the cloud, then return to their work weeks or months later. More importantly, such an archive could be published together with a refereed journal article, so that readers are able to interactively explore and reproduce the published simulation results, using the same version(s) of the code and its dependencies, already compiled in exactly the same way. Collaborators will also benefit from this ability to share a complete simulation environment "in a box". Commercial companies [31,32] are beginning to offer services along these lines to other communities.

The Browser is the UI

Our vision is that the web browser will become the ubiquitous user interface (UI) for scientific computing. This ambitious goal has become viable very recently, due to the emergence of powerful, standardized technologies, including HTML5 [33], CSS [34], JavaScript [35-37] and scalable vector graphics (SVG) [38]. There are numerous technologies for scientific visualization, which build on these standards.

In the near term, we will use JavaScript emulation of the X-windowing protocols to support scientific codes running on Linux in the cloud. Of platforms which can run X-based UIs in the browser, GTK+ [39] has the most sophisticated system, called Broadway [40]. For example, the open source QTHTML project [41] uses Broadway to enable Qt widgets in the browser. In the long-term, we will use web technologies to develop custom UIs for scientific codes.

Seamless Legacy

We respect the existing workflows of computational scientists, so our vision includes support for both command line and web based UIs. Also, any code we provide via the cloud will also be available for use on desktop and laptop computers.

APPLICATION CONTAINERS

We briefly explain application containers and how they differ from virtual machines. A hypervisor fakes the messages and addresses that a computing kernel expects to see from a physical computer. When the kernel is loaded into memory, the hypervisor runs it like any other process. There are two kinds of hypervisors: hosted and native. When you rent a virtual machine from Amazon Web Services, your virtual machine is running on a native hypervisor. This means the hypervisor is the "kernel" for the computer.

Virtual machines (VM) running on your laptop are running on a hosted hypervisor (e.g. VirtualBox [14], Parallels [42], VMWare [43], etc.). VMs do a lot of work to maintain a complete abstraction of a physical machine. This is inefficient and, it is unnecessary for most processes.

Containers are a complete abstraction of all the relevant resources used by the vast majority of programs. Instead of emulating the computer, LXC (LinuX Container) [44] isolates kernel resources for a collection of processes (container). Essentially, LXC is a lightweight, hosted hypervisor without need for emulation. Vagrant [13] is a program for configuring headless VMs and containers. Like Vagrant, Docker [12] is a productivity enhancer for LXC. It assumes you are running on Linux, which means you will need Vagrant to boot a Linux VM on your laptop before you can run a container with Docker.

We have chosen to use Vagrant as the deployment framework for RadTrack VMs. This greatly simplified the work we had to do managing VirtualBox VMs. Once Vagrant is installed, an end-user need only type three commands to start RadTrack:

```
$ vagrant init radiasoft/radtrack
$ vagrant up
$ vagrant ssh -- -Y vagrant-radtrack
```

The first command prepares a configuration file in the current directory. The second command downloads the VM 'box' to the user's computer, unpacks it, and boots a unique instance of the VM. The third command connects the user's terminal window to the 'headless' VM using ssh. The vagrant command knows how to find the appropriate port, which includes automatic collision correction. This is very convenient for developers who will certainly have multiple VMs running, possibly for different projects.

ACKNOWLEDGMENT

This work is supported by the US DOE Office of Science, Office of Basic Energy Sciences through Grant No. DE-SC0006284.

REFERENCES

[1] S. Seung, G. Andonian, M.A. Harrison, S. Wu, T. Shaftan, D.L. Bruhwiler, "A user-friendly, modular simulation tool for laser-electron beam interactions," NA Part. Accel. Conf., MOPBA17 (2013).
[2] RadTrack; https://github.com/radiasoft/radtrack
[3] Qt; http://www.qt.io
[4] PyQt; http://riverbankcomputing.com/software/pyqt
[5] M. Borland, "Elegant: A flexible SDDS-compliant code for accelerator simulations," APS Technical Report, LS-287 (2000).
[6] O. Chubar and P. Elleaume, "Accurate and Efficient Computation of Synchrotron Radiation in the Near Field Region," Proc. European Part. Accel. Conf., 1177 (1998).
[7] O. Chubar, P. Elleaume, S. Kuznetsov, A. Snigirev, "Physical Optics Computer Code Optimized for Synchrotron Radiation," Proc. SPIE 4769, 145 (2002).
[8] O. Chubar, A. Fluerasu, L. Berman, K. Kaznatcheev and L. Wiegart, "Wavefront propagation simulations for beamlines and experiments with SRW," J. Phys.: Conf. Ser. 425, 162001 (2013).
[9] SRW; https://github.com/ochubar/SRW
[10] S. Reiche, "GENESIS 1.3: a fully 3D time-dependent FEL simulation code," Nuclear Instruments and Methods A429, 243 (1999).
[11] GENESIS 1.3; http://genesis.web.psi.ch
[12] Docker; https://www.docker.com
[13] Vagrant; https://www.vagrantup.com
[14] VirtualBox; https://www.virtualbox.org
[15] Screenflick; http://www.araelium.com/screenflick
[16] BB FlashBack; http://www.bbsoftware.co.uk/BBFlashBack_FreePlayer.aspx
[17] M. Borland, "Getting started with SDDS"; http://www.aps.anl.gov/Accelerator_Systems_Division/Accelerator_Operations_Physics/manuals/GettingStartedWithSDDS/GettingStartedWithSDDS.pdf
[18] M. Xie, "Design Optimization for an X-Ray Free Electron Laser Driven by SLAC Linac," Proc. Part. Accel. Conf., 183 (1996).
[19] J. Alda, "Laser and Gaussian Beam Propagation and Transformation," *Encyclopaedia of Optical Eng.* (Marcel Dekker, New York, 2002).
[20] F. Pampaloni and J. Enderlein, "Gaussian, Hermite-Gaussian, and Laguerre-Gaussian beams: a primer" (2004); http://arxiv.org/abs/physics/0410021
[21] A.E. Siegman, *Lasers* (Univ. Science Books, 1986).
[22] Wikipedia page on paraxial laser modes; https://en.wikipedia.org/wiki/Gaussian_beam
[23] Synergia 2.1; https://web.fnal.gov/sites/Synergia/SitePages/Synergia%20Home.aspx
[24] D.P. Grote, A. Friedman, J.-L. Vay, I. Haber, "The WARP Code: Modeling High Intensity Ion Beams," AIP Conf. Proc. 749, 55 (2005).
[25] J.-L. Vay, D.P. Grote, R.H. Cohen and A. Friedman, "Novel methods in the Particle-In-Cell accelerator Code-Framework Warp," Comput. Sci. & Disc. 5, 014019 (2012).
[26] WARP; http://warp.lbl.gov
[27] WARP source; https://bitbucket.org/berkeleylab/warp
[28] M. Sanchez del Rio, N. Canestrari, F. Jiang and F. Cerrinac, "SHADOW3: a new version of the synchrotron X-ray optics modelling package," J. Synchrotron Radiation 18, 708 (2011).
[29] SHADOW3; https://github.com/ncanestrari/shadow3
[30] Open source RadTrack development on Github; https://github.com/radiasoft/radtrack/issues
[31] Terminal; https://www.terminal.com
[32] rescale; http://www.rescale.com
[33] HTML 5.1; http://www.w3.org/TR/html51
[34] CSS; http://www.w3.org/Style/CSS
[35] JavaScript; http://en.wikipedia.org/wiki/JavaScript
[36] JavaScript – MDN; https://developer.mozilla.org/en-US/docs/Web/JavaScript
[37] ECMAScript; http://www.ecma-international.org/memento/TC39.htm
[38] SVG 1.1; http://www.w3.org/TR/SVG
[39] GTK+; http://www.gtk.org
[40] Demo of the GTK Broadway backend; https://www.youtube.com/watch?v=fr8eo4RlPw4
[41] QTHTML; https://github.com/Etrnls/qthtml
[42] Parallels, https://www.parallels.com
[43] VMware, http://www.vmware.com
[44] LXC, https://linuxcontainers.org

ISBN 978-3-95450-168-7