Using Networked Data Services for System Analysis and Monitoring

Justin Owen

Irondequoit High School

Advisors: Mr. Rick Kidder, Mr. Colin Kingsley, Mr. Michael Spilatro

Summer 2009

Abstract:

By developing and connecting instruments to a networked data service rather than monolithic applications, it is possible to make the instruments' data available to an entire network. This can enhance laser operation capabilities by allowing data from multiple instruments to be combined in real time in a computer program. This is referred to as orchestration. This project demonstrates orchestration by combining data from two operations diagnostic services into one client program. The service programs are the existing Time-Expanded Single Shot Autocorrelator (TESSA) and a spectrometer service that both produce data on the network. The focus program is a client application that is coded in Java and a middleware language called Internet Communication Engine (ICE) to receive data from the OMEGA EP front-end laser diagnostics.

Background:

The Laboratory for Laser Energetics (LLE) currently has numerous diagnostic and measurement systems on the OMEGA EP laser that collect data when it is preparing for and firing a shot. When the data is collected, it is normally stored in an archive of HDF4 data files with an .hdf file extension. This format is accepted by the scientific community and is extensible to easily provide for the many differing types of data found on the system. While it is very efficient to store archival data in this type of file, which can be referenced later, it is very inefficient for real time operations. The data would be significantly more useful if it could be monitored in real time (i.e., during shot preparation) without the overhead of storage to the file servers. For that purpose, these and some other of the OMEGA diagnostics have been associated with ICE, allowing for data from the diagnostics to be sent over the network in real time.

ICE is an object-oriented middleware platform designed to simplify networked programs, and is compatible with C++, Java, Python, and several other programming languages. ICE provides facilities for developing unified interfaces that any of the supported programming languages can interact with, as well as providing necessary network arbitration. ICE also provides the IceGrid (see Figure 1) locator service, which simplifies the task of finding the ICE objects that one wants to access on the network, and the IceStorm service, which handles multicasting data to other ICE enabled applications.

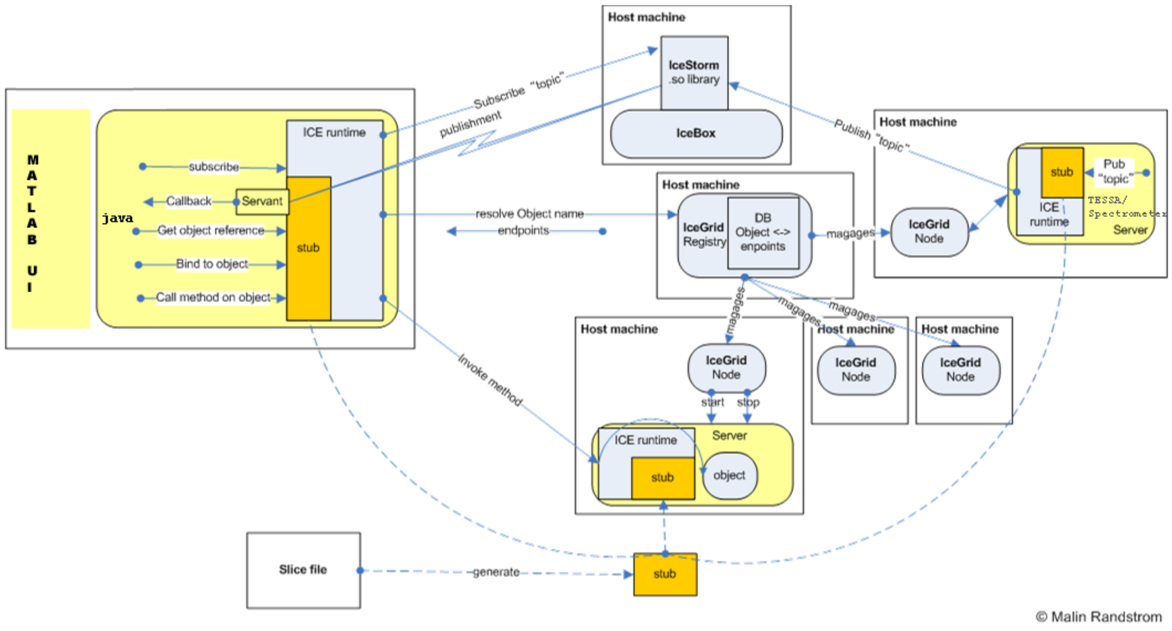
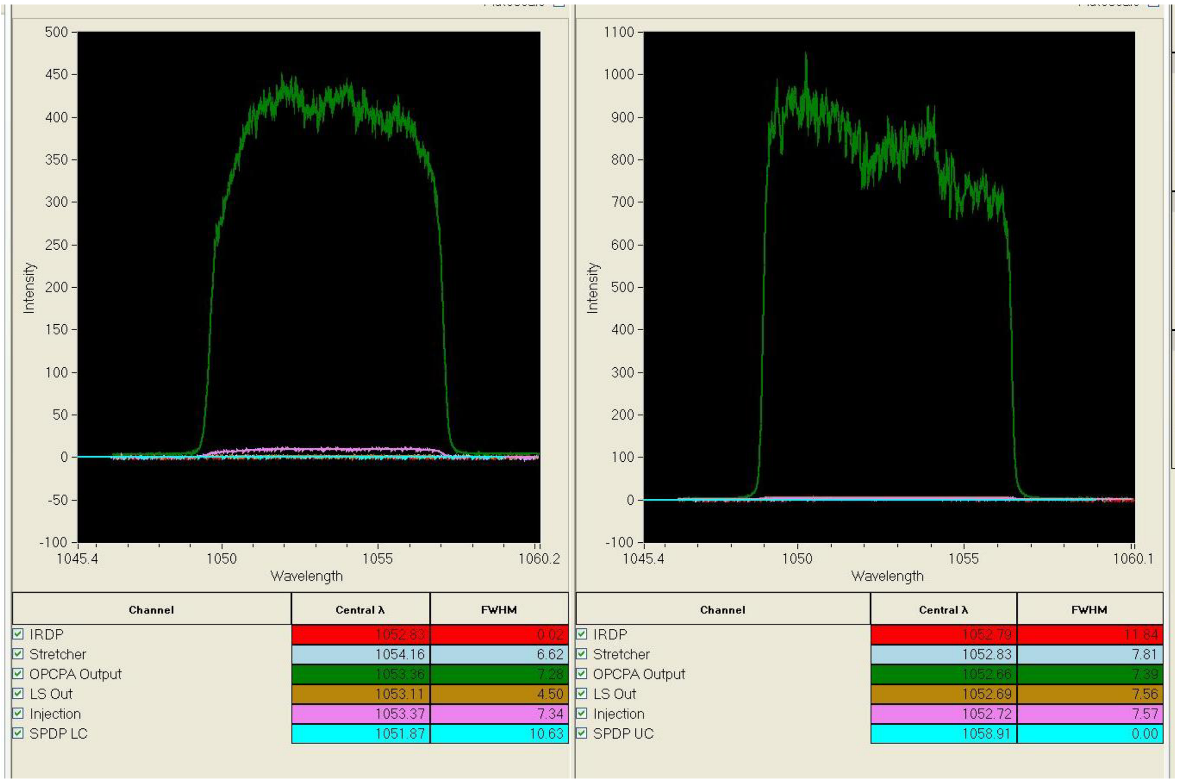


Figure 1. Depiction of ICE services in relationship to Java and the Matlab User Interface. Drawing is a revised IceGrid architecture drawing by Malin Randstrom, http://upload.wikimedia.org/wikipedia/en/5/5a/ICEgrid.png.

Among its diagnostic systems, the OMEGA EP laser currently has two spectrometers for measuring the laser spectrum on beamlines 1 and 2 (see Fig. 2),



Channel	Central λ	FWHM
☑ IRDP	1052.83	0.02
☑ Stretcher	1054.16	6.62
☑ OPCPA Output	1053.36	7.28
☑ LS Out	1053.11	4.50
☑ Injection	1053.37	7.34
☑ SPDP LC	1051.87	10.63

Channel	Central λ	FWHM
☑ IRDP	1052.79	11.84
☑ Stretcher	1052.83	7.81
☑ OPCPA Output	1052.66	7.39
☑ LS Out	1052.69	7.56
☑ Injection	1052.72	7.57
☑ SPDP UC	1058.91	0.00

Figure 2 Spectrometer diagnostic display for the Omega EP laser system beamlines 1 and 2. Data recorded by the spectrometers on a typical shot, from beamline 1 (left) and beamline 2 (right). Spectra are superposed for various channels, corresponding to different positions along the laser beam paths.

and the Time-Expanded Single-Shot Autocorrelator (TESSA), which measures the short pulse (~1.0 ps) laser's pulse width (see Fig. 3).

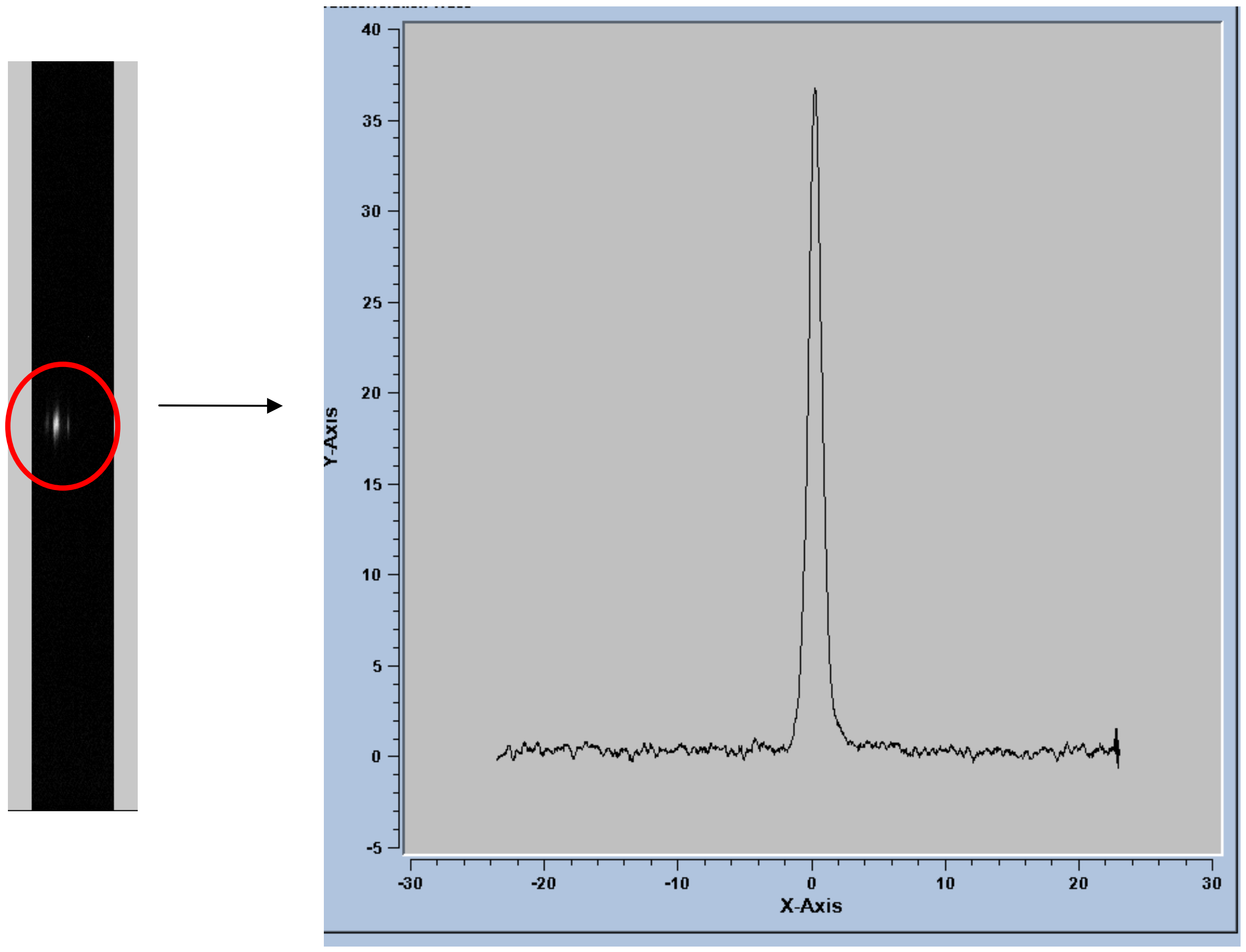


Figure 3 TESSA display of short pulse data captured from the upper compressor fed by Omega EP beamline 2. The image on the left is converted to the graph on the right, where the x axis corresponds to time. The full-width at half max is used to determine the temporal duration of the laser pulse.

Analysis of this data, combined with characterization details on system components, can provide predictive capabilities for the laser output. As such the spectrometers and TESSA have been configured to send the data they collect in real-time to a central IceStorm server, which can send the data to any programs on the network that "subscribe" to the instruments' “published” ICE events. IceStorm is a publish / subscription service provided as part of ICE.

Purpose:

The purpose of this project is to create an example program that can use IceStorm to combine the real-time data from multiple diagnostics to create useful information that can be viewed in real-time by those working with the OMEGA EP laser. The .hdf files in which the data is currently stored after collection cannot be used until the diagnostics have finished writing the files and so useful analysis on the data can only be performed after the collection is over. This leads to substantial overhead on the file systems by producing many pre-shot files that are not of archival quality. These files require significant resources to manage to ensure data integrity and cleanup after the files are no longer needed. This burdens developers and operation staff who are responsible for the diagnostic and laser systems. By providing the data service, file storage and overhead is eliminated and analysis can be performed in real-time during the set-up for a shot.   The nature of IceStorm also means that multiple programs can use the real-time data feeds simultaneously. This provides flexibility where tasks can be broken between many client utilities.

In addition to orchestration, the project demonstrates the potential gains of developing a Service-Oriented Architecture on the LLE network. A Service-Oriented Architecture would consist of a set of “services” on the network with well-defined interfaces, so that any program that knows the interface for a service can use it; TESSA and the spectrometers sending data through IceStorm provide examples of services that produce data. A Service-Oriented Architecture for LLE would allow applications to interact with each other easily over the network, enabling programmers and use applications to create new, useful information.

Final Product:

The final product is a set of Java classes and ICE files for using the data services from the spectrometers and TESSA in a few configurations. The code is designed to be used by frontend programs such as Matlab to access the data being received from the services that run relevant analysis on the data.

The program was written to retrieve data from one spectrometer, both spectrometers, TESSA, or all three simultaneously; each program also has methods to provide just a subset of the available information it receives (e.g., just the frequency/intensity data from one spectrometer) depending on the service interface.

The spectrometer program has functionality to return a three-dimensional array of values, providing a graph of frequency and intensity for each channel on the spectrometers (see Fig. 2). The TESSA program returns an array of values representing the intensity at each point in time, a value representing the "timescale" (the difference in time between each point on the array), a value representing the "offset" of the pulse, or the width of the pulse at half its maximum intensity. It can also return a matrix with all ordered values.

Because the code was initially intended for use with Matlab, which is not multithreaded and does not support asynchronous event notification, a satisfactory interface that would allow Java to call Matlab functions could not be developed. The analysis function in Matlab must manually retrieve the data from the Java data handler program (see IceClient Java class below). Ideally, the program would be able to call an analysis function in Matlab as soon as it received data. The Java program, therefore, “holds” the most recent set of data received from the spectrometer and TESSA services until Matlab calls the appropriate Java method to retrieve a copy of the data.

The code also includes an “IceClient” Java class that contains the code used to actually interface with Ice and IceStorm; the programs extend this class. This fortunately makes writing code for new services relatively simple, since any new code can also use the IceClient class instead of requiring unique server code to be written. A readme file is included with the code with recommendations on writing similar code for other IceStorm services; it also includes instructions on using the code.

Next Steps:

Future work on this project should first include modifying or creating an analysis program to take the spectrometer and TESSA outputs passed on by the program described here. Such an analysis program should be able to display the data and use it to provide predictions on laser pulse outputs of the OMEGA EP beam. Such a program could possibly be used to validate system configurations or automate part of the calibration process for the OMEGA EP laser. This could be combined with data taken on a previous shot to adjust the laser parameters to produce more optimal shots. This process saves time and labor in the OMEGA EP control room by automating sometimes tedious shot configurations.

More future work would involve connecting other instruments in the OMEGA EP laser and LLE to ICE services, so that more services can be integrated and used over LLE’s internal network.

Acknowledgments

I would like to thank Richard Kidder for giving me the opportunity to work on this project. I would like to thank Colin Kingsley and Mike Spilatro for advice regarding the scope and applications of this project. I thank Dan Gresh for doing some unpublished initial work on the project, and Ralph Russo for advice on programming in Ice. Finally, I would like to thank Stephen Craxton for coordinating the high school program.