CBASS - BNL's Python-based Synchrotron Beam line- and Experiment-Control System

John Skinner, Robert Sweet, Howard Robinson
Biology Department
Brookhaven Nat'l Lab.

Abstract

For over a decade, it has been recognized that efficient use of a Macromolecular Protein Crystallography beamline demands software that is easy to use. While designing an intuitive interface to the experiment presents significant difficulties, the implementation of the underlying control code is complicated by the ever-changing hardware that it controls. Hardware has traditionally been limited to beamline motors, scalers, diffractometers, and X-ray detectors. New challenges for the designers of integrated software are arising as sample changing robots, the need for remote observation or operation, and associated project-management database systems are developed to meet the demands of high-throughput crystallography. It is not sufficient for the software that controls the frequently changing hardware and data collection protocols simply to work in a flawless manner, it must be easy to understand, expand, and modify.

With this in mind, we have developed CBASS (Crystallography at Brookhaven Acquisition Software System), a user-friendly Python-based system that integrates data collection and beamline control, with the option of incorporating a sample-mounting robot and project-management relational database system. Its modular design, combined with the inherent readability and relative ease of use of the Python programming language, minimizes installation and adaptation complexities often encountered in a dynamic environment.

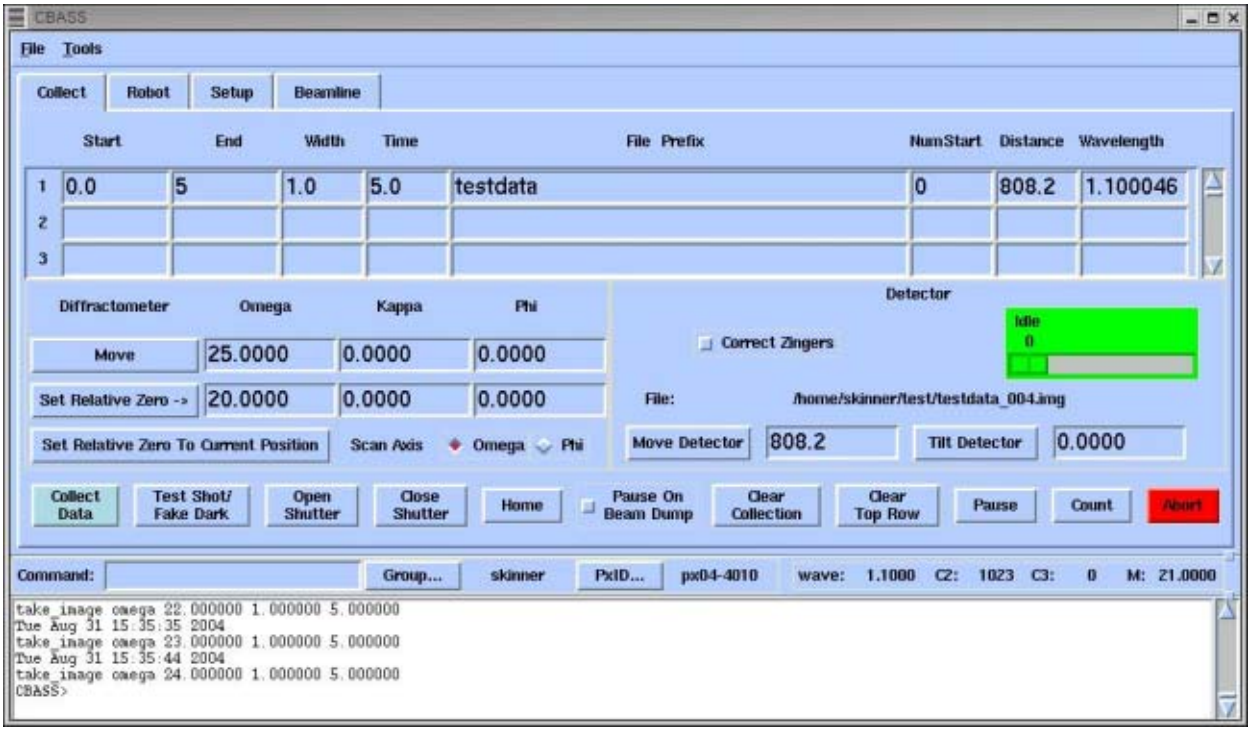

Figure 1.

The Display Scheme

The CBASS window is implemented as a "notebook" widget with between one and four "pages" that can be selected by clicking on tabs named "Collect", "Robot", "Setup", and "Beamline". The pages that can be displayed are controlled by environment variables set in the CBASS configuration file. These are "has_robot", "has_screening", and "has_beamline". Figure 1 shows the main window with the four page tabs available as of this writing. This is how the main window would look if all of the page control environment variables were set to "1".

The GUI is a client that constructs commands and sends them to the CBASS command server. These same commands can be executed by typing them directly into the "Command" window. Group and project identifiers can be selected by pop-up selection boxes by pushing the appropriate buttons next to the command line. This information is used for the BNL PxDB experiment tracking database. Counter readouts and monochromator wavelength are also displayed in this window pane. The white scrolled window displays output from the command server.

The “Collect” Page

In about the middle of the Collect page on the left hand side, two sets of values are presented for the diffractometer angles (Omega, Kappa, Phi). The first set is the current position, in this case 25.0, 0.0, and 0.0. If one changes any of these values, and then pushes the "Move" button, the axes will move to the new position. The second row of numbers is referred to as the "Relative Zero" or "Datum" position. This is the starting position for the data sweeps defined in the scrolled table at the top. If the "Collect Data" button was pushed in the screen shot above,

five images of oscillation width 1-degree would be taken from Omega = 20 degrees to Omega = 25 degrees. These would be 5 second images resulting in data file names testdata_000.img through testdata_004.img.

One can change the relative zero by changing the text fields and pushing the "Set Relative Zero" button, or one can set the relative zero to the current position by pushing the "Set Relative Zero to Current Position" button.

The "Detector" box at the right allows one to change the crystal-to-detector distance as well as the detector tilt (2**2**). To change one of these, edit the textfield, then push the appropriate button. If the exposure times are greater than one minute, one may need to select "Correct Zingers". For two-minute exposures with this option selected, the software would take two one-minute images and combine them while throwing away outliers caused by things like cosmic rays.

There are two ways to take images. Normal collection, which always forces proper dark image corrections, is started with the "Collect Data" button. Collection proceeds as was described for Figure 1.

Test shots, which take two-second darks in order to save time, can be collected by pushing "Test Shot/Fake Dark". These images are suitable for visual screening. Realtive Zero is ignored, and the software will simply take a single image at the current position for the desired frame width and exposure time. If the "Width" field = 0, a "still" image will be taken.

Other buttons below the Diffractometer and Detector frames allow for diffractometer shutter control, homing the diffractometer in case of a power cycle of the instrument, pausing of the data collection, and aborting the data collection.

If the beamline is to be left unattended for a significant period of time, the user can activate the "Pause on Beam Dump" feature. With this enabled, the software will monitor a remote file that displays the status of the x-ray source, provided and maintained by our facility, the National Synchrotron Light Source. When the software detects a beam dump, it will wait for the beam to come back, pause for a beamline-specific period of time, run a beamline-specific beam optimization routine, then resume data collection starting with a retaking of the image that was in progress when the beam dumped.

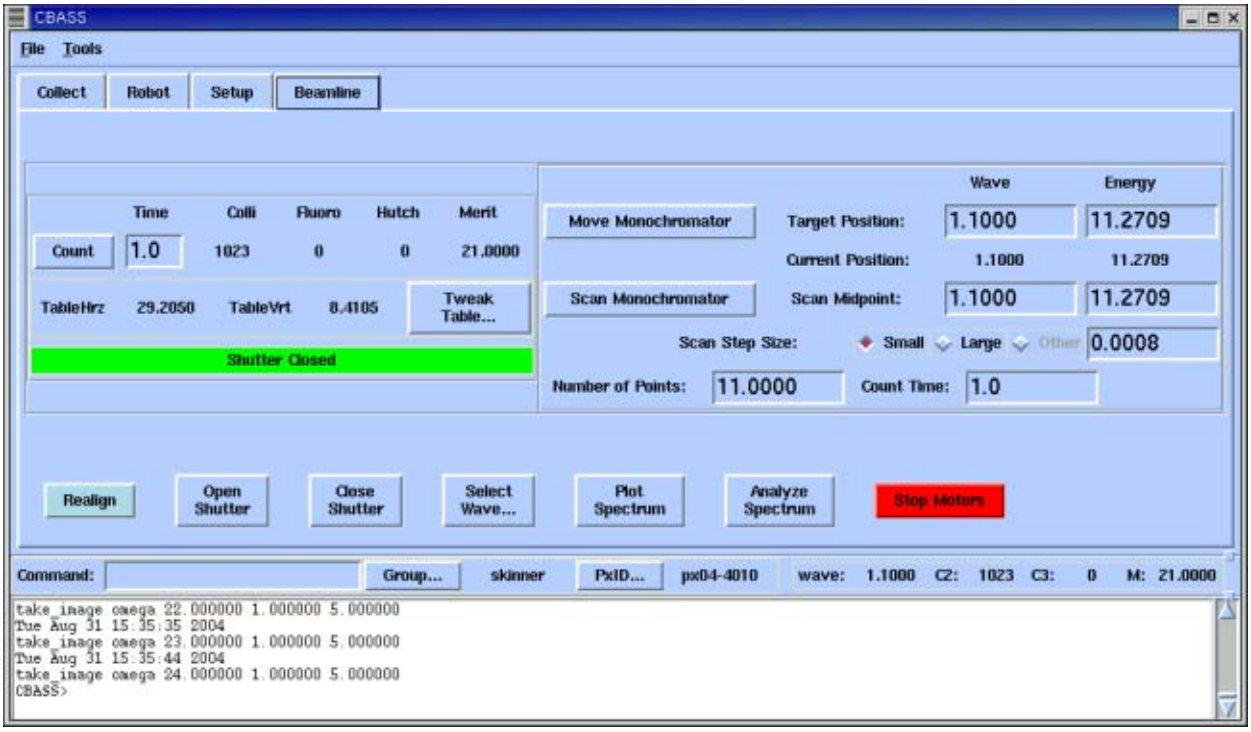

Figure 2. - Beamline Page

The “Beamline” Page

Basic beamline control, such as monochromator scans and frequently used beamline alignments, are controlled from the "Beamline" page (Figure 2).The intent of this page is to provide an interface to the beamline components and functions used during typical experiments. More complex beamline operations, such as mirror adjustments, which are normally performed by beamline staff are handled with BNL's GrEpx software, which provides full access to beamline motors.

Pushing the "Count" button will display the counter readings at the collimator, fluorescence detector, and an additional upstream counter. The diffractometer table can be adjusted with a "Tweak Box". Monochromator settings and scan parameters are controlled in the right hand side of the window. Buttons at the bottom of the page allow for beamline-specific realignments, shutter control, and monochromator scan analysis.

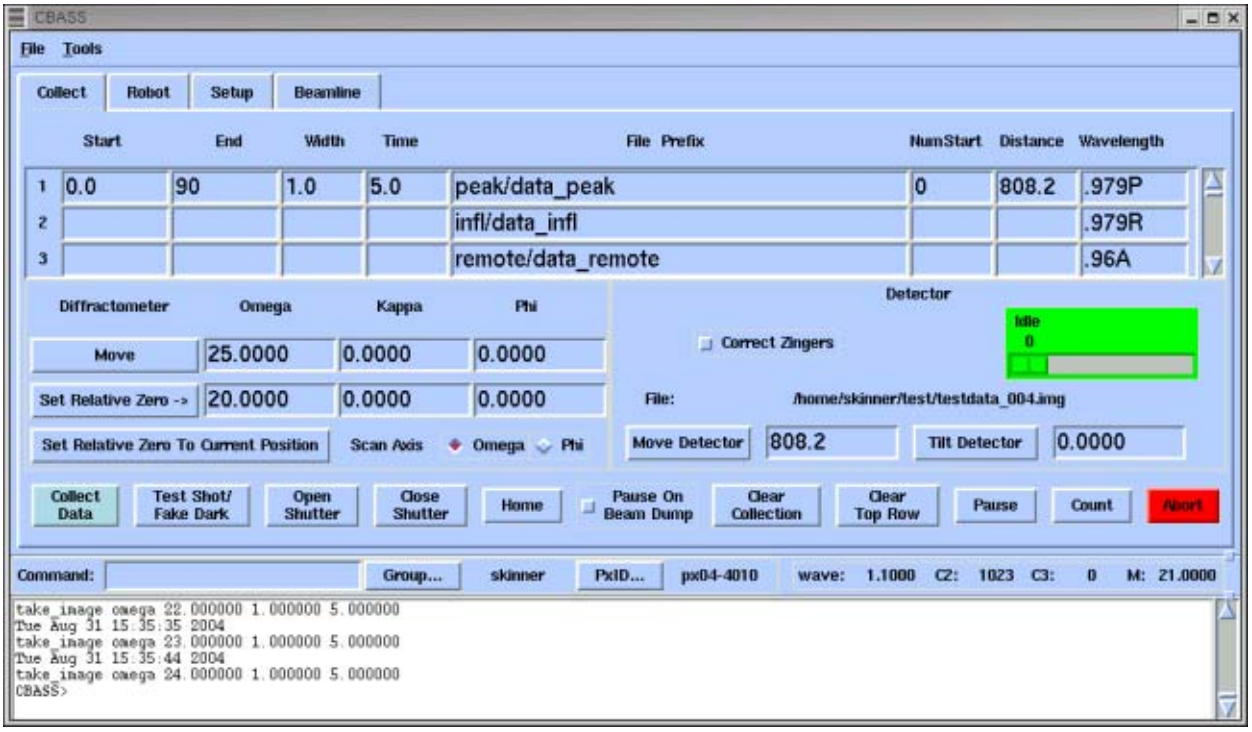

Figure 3. - Automatic MAD

Multiwavelength anomalous diffraction (“MAD”) experiments can be programmed to run in an automatic fashion as seen in Figure 4. There are four character codes that one can append to the wavelength value to cause the program to take action before taking the data sweep. In the top row of Figure 3, ".979P" (P=Peak) will cause the software to scan the monochromator with a midpoint wavelength of .979 using the scanning parameters set in the Beamline page. When the scan is completed, an analysis will run and the monochromator will move to the peak of the scan, then the data sweep will proceed. In the second row, ".979R" (R=Rising edge) will cause it to scan and move to the rising edge before taking the data sweep. "I" for "inflection point" will work the same as "R".".96A" (A=Absolute) will move the wavelength to .96. Data collection will start without performing a monochromator scan.

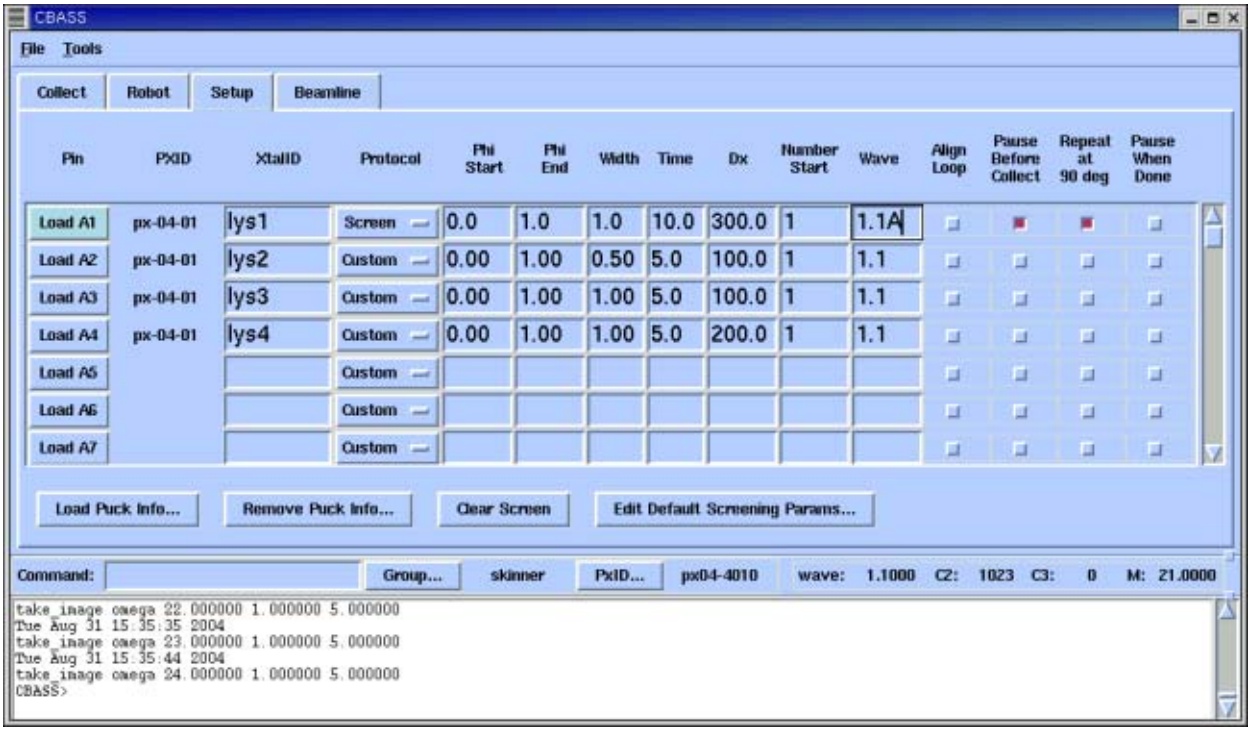

Figure 4. - Screening or Setup Page

The “Setup” Page for automated robot use

The "Setup" page (Figure 4) was designed originally to be used with the BNL sample mounting robot, but it is also finding extensive use with screening and collection on specimens sent in for "FedEx" data collection. The "Pin" buttons A1-D16 map to the robot dewar which contains four crystal pucks (A-D) which each contain 16 pins. Information describing the contents of each puck and the desired actions on each crystal are loaded into PxDB with a web-based interface (Figure 5). Pushing the "Load Puck Info" button will pop up a selection box that allows the operator to select pucks belonging to the current group. Selecting a puck will fill in the chosen puck position (A-D) with information and parameters found in PxDB. Pushing a pin button will load information into the "Collect" page table. When collection is underway on items loaded in this fashion, the robot will mount samples from the appropriate puck/pin positions.

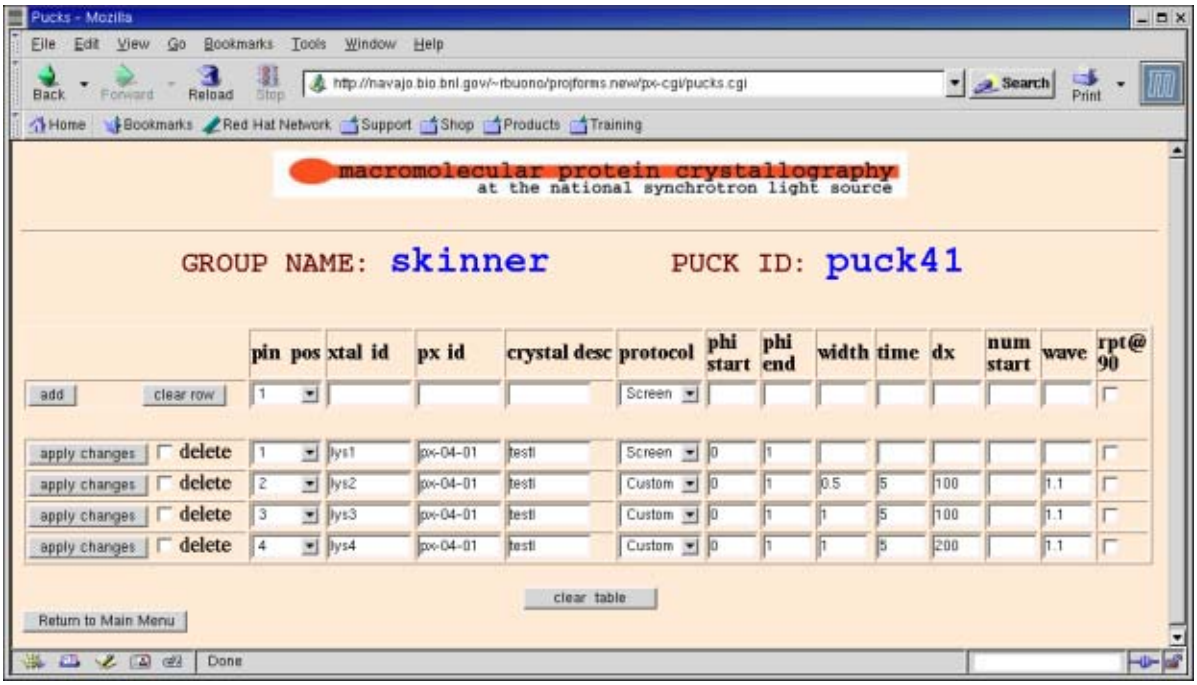

Figure 5 - Puck Description Web page

Directories of the form "./[PXID]/[XtalID]/sweep_number/ will be created automatically for the resulting data images to be collected. "Sweep number" is determined by querying PxDB for previous data sweeps performed on the XtalID with the current PXID. If the "Protocol" option menu is set to "Screen", then parameters will use defaults established at the beamline. These can be changed by pushing "Edit Default Screening Params...".

The “Robot” Page

As of this writing, the "Robot" page was set up for simple mounting and unmounting of individual samples. The "Setup" page provided everything necessary to drive the robot during data collection.

The Tools Menu

A variety of features can be run from the "Tools" menu on the main menubar.

AutoAdxv

Beamlines with a detector from Area Detector Systems Corp. can execute ADSC's "ADXV" program with the "autoload" option. In this mode, ADXV will display images as they are collected.

AutoMax

The CBASS "automax" facility can be used to perform automatic beamline alignments during data collection. This tool prompts for the frequency in minutes at which the user would like optimizations to be run. When it is time for automax to perform an optimization, it pauses data collection, runs the beamline-specific realignment, then resumes collection

CrystalView

The "Crystal View" tool employs our Axis™ video server to display a live image of the crystal. On beamlines with the appropriate hardware, this can be used to center the crystal in response to the user clicking on the image.

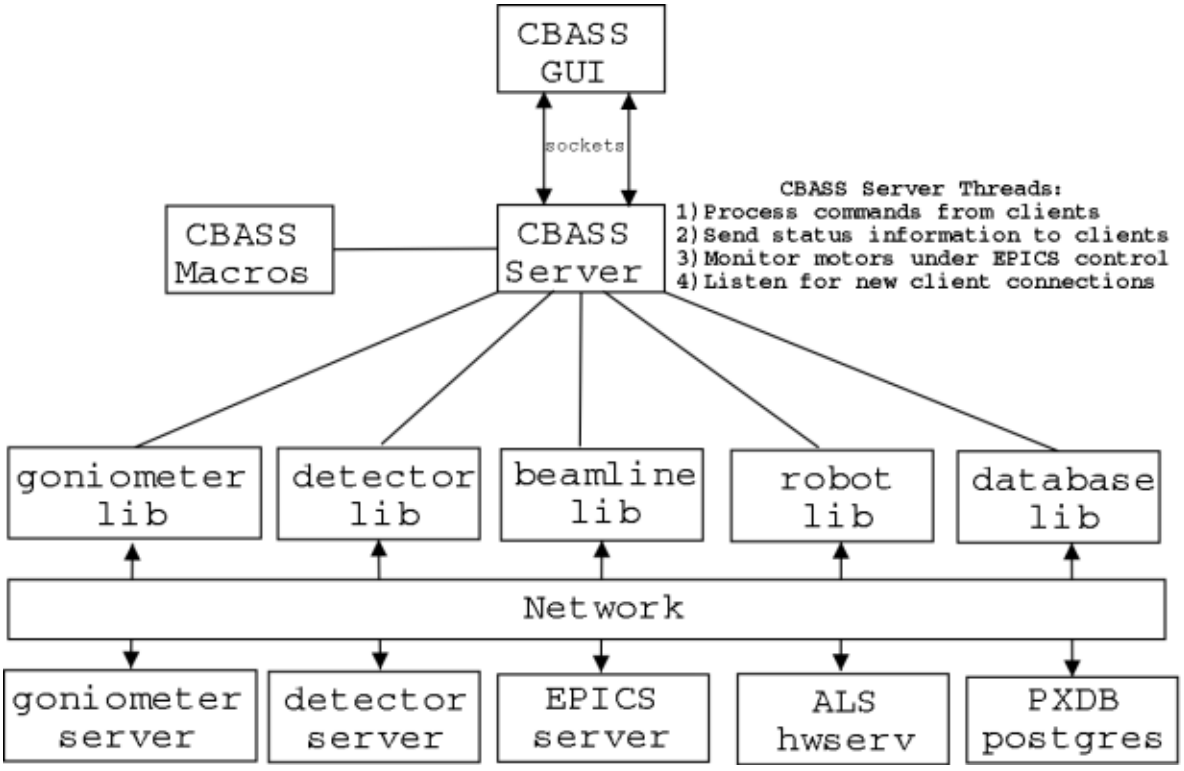

Figure 6 - CBASS System Architecture

System Architecture and Configuration

The software is run under Linux at the NSLS. Since Python code is highly portable, it should be possible to run it under other operating systems as well.

CBASS is set up as a client/server system in which the GUI communicates with the CBASS control server over TCPIP sockets (Figure 6). Multiple clients can communicate with the server. This has been employed at NSLS beamline X25 where a Java applet running on a Windows-based laptop runs as a CBASS client to facilitate in-hutch goniometer control, while the main CBASS GUI runs at the user end-station.

The CBASS server is a multi-threaded program with individual threads responsible for processing commands from clients, sending status information to clients, reading the status of beamline motors that are under EPICS control, and listening for new client connections. One of the strengths of CBASS is that it is a relatively small and readable program consisting of roughly 6K lines of Python code. The server code is broken up in a modular fashion, with separate python modules written for diffractometer, detector, beamline, database, and robot control. This approach simplifies adapting CBASS to different hardware since only small individual modules would need to be coded.

BNL's lower level control is implemented with servers. This was done to isolate modules such as "goniometer_lib" and "detector_lib" from hardware differences. This was beneficial because our older Nonius CAD4 diffractometers and Brandeis detectors could be controlled only by SGI's, while we preferred to run our newer Compumotor-based diffractometers and ADSC detectors under linux. By writing servers with identical communication interfaces we arranged that the upper level code doesn't have to change if we move or replace hardware. The beamline motors are served through the EPICS Channel Access server. Database access is accomplished by Python's interface to the Postgres database server. We control the sample mounting robot by communicating with a robot hardware server written at the ALS.

Beamline-specific routines, such as those used for automatic beam dump recovery and beam alignments, are found in the "cbass_macros.py" file in the CBASS configuration directory local to each beamline. Beam line staff, or even users depending on file-permissions, can add or modify routines and load them into the running system by typing "reload macros" in the CBASS command line.

The program can be tailored to different configurations through the editing of environment variables set in "cbass_env.txt" in the CBASS configuration directory. As previously mentioned, variables are set to indicate the use of the robot and project tracking database.

Most CBASS defaults can be set in the "cbass.site" file local to each beamline. This command file sets common flags and parameters, such as whether or not images should be binned or counter intensities should be monitored during data collection.

Summary

Only a few years ago, almost all Protein Crystallography experiments performed at synchrotron beamlines involved a user group coming to the beamline to manually mount and collect data on several crystals per day that were usually associated with a single project. Higher beam intensities, as well as advances in techniques related to expression, purification, crystallization, and other fallout from the Human Genome Project, have changed the way many beamlines are used. It is now not unusual for a hot beamline to collect on dozens of crystals in a day that belong to several different projects from multiple user groups. Automation aids, such as robotic sample changers and project management databases, have been developed to cope with the increase in crystals and intensity, and new data-collection protocols have been designed to make the most efficient use of all resources. The dynamic state of today's PX facilities requires that the software controlling everything be extremely flexible and adaptable to new components

and ideas. Brookhaven's CBASS system has been developed in response to these demands. It's modularity combined with the relative ease of use of the Python programming language has resulted in a robust and easily extensible software package for Protein Crystallography work stations.