DISPLAYING AN EPICS WAVEFORM DATA AS AN IMAGE IN PYTHON/TKINTER

N. Yamamoto,* KEK, Tsukuba, Ibaraki, 305-0801, JAPAN

Abstract

Scripting languages such as Python and SAD together with Tk widgets are successfully used in the KEKB control system. These tools enhances flexibility and usability of the system. Its performance satisfies most of requirement in the system. However, some application, such as an image display on the operator screen, may hits their performance limit. This performance limits can be easily overcome combining a Tk widget and an EPICS CA callback in the C language level rather than interpreter level. Use of this approach enable us to update an image on a X window screen at 20 frame per second or higher through EPICS channel access.

EPICS AND SCRIPTING LANGUAGES

EPICS is a software toolkit to construct wide variety of control system based on network distributed architecture [1]. EPICS provides generic software for building and running operator user interfaces. These software, called EPICS CA clients, are configurable through a configuration file and fulfill most of needs in a control system.

However, some applications requires the functionality beyond the ability these tools can provide. Then we need to develop an application to satisfy this requirement. Scripting languages, such as Tcl/Tk, Perl, Mathematica, PV-wave, Python and SAD, are widely used in the EPICS collaboration[2]. These languages are used for prototyping these applications, however, in some cases, the applications were written in these scripting languages.

Python, Tcl/Tk /BLT widget and EPICS CA

Python is an object oriented scripting language developed by G. von Rossoum. One of its rich extension modules is Tkinter.py which enable us to access Tk widget from Python programs. Tk toolkit includes a widget for presenting photo image on display. An image on this photo image widget can be dynamically modified supplying pixel data of new image.

EPICS channel access interface for Python was developed by author and is used in the KEKB control system[4] and will be used in J-PARC project[5]. When the EPICS CA module receives data from CA, binary channel data is converted to Python object in the callback routine [Fig. 1].

BLT[6] is an extension library Tcl/Tk widget. Among the functionality provided by BLT, a graph widget and a vector widget are frequently used in the applications fro KEKB operation system. Pmw(Python Mega Widget) [7] is used to access BLT widgets from Python interpreter.

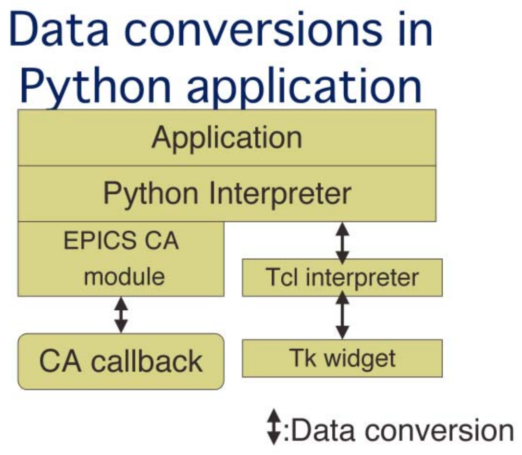

Figure 1: Binary data from CA callback is converted to Python objects. When passing data from Python to Tk widget, Python object representing an array of floating point number is converted to a string representation, then converted to binary form data in Tk widget by Tcl interpreter.

Since Tk widget implementation strongly depends on Tcl interpreter, Python Tkinter modules actually pass command as a string to Tcl interpreter embedded in Python interpreter[Figure 1]. This approach ensure that we can access all functionality of Tk widget from Python.

Overhead caused by these data conversion can be ignored for scalar data in a usual operation. However, overhead dominate CPU usage for long waveform data at reasonably high update rate.

Data conversion overhead from binary data to Python is much smaller than that from Python object to string data. So if we can avoid data format conversion related to the string format, we can expect great improvement of performance[Figure 2].

CONNECTING CA CALLBACK DIRECTLY TO A TK WIDGET

To update PhotoImage widget or Vector widget on EPICS CA event, we need to

1. provide a way to relate EPICS channel to a photo image widget or BLT vector object, and

* noboru.yamamoto@kek.jp

2. setup callback to transfer data in EPICS CA buffer to Tk photo image structure or BLT vector object.

These functions take three parameters, channel id , interpreter and widget name. These python functions are generated from C functions with same names using a generic wrapper generation tool, SWIG(Simple Wrapper Interface Generator).

These functions also setup EPICS CA callback using predefined C callback routines. These C callback functions copy data from EPICS CA buffer to the data area of widget structure, Tk_PhotoImageBlock and Blt_Vector. This callbacks routines also takes care of data conversion if necessary.

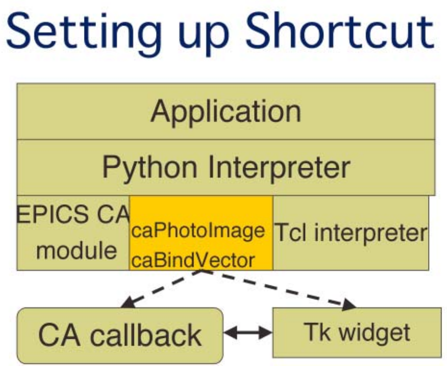

Figure 2: caPhoto Image module setup a direct CA callback link to Tk Widgets.

PERFORMANCE

On powerbook G4 12”, 800MHz Power PC with 384MB memory. We don’t see any difficulty updating a photo image widget connected 16K size short integer array at 20Hz. Both ioc shell and Python program were running on same machine. At that time, an ioc shell process uses 20 % of CPU time. Python program displaying three Photo images uses 50 % of CPU time.

CONCLUSION

We have developed a software which improves performance of displaying an array data on the operator screen even using the scripting language. Using proper tools like a SWIG, the development time was greatly reduced.

Current implementation assumes data type of the array. In the future version of software, any type of array supported in EPICS may be supported.

This software uses internal data structure in Tcl/Tk and BLT toolkit. This software may strongly depend of the version of these base libraries. However, the difference of the structure can be easily fixed, because these structures had been used longtime already and are already mature stage.

REFERENCES

[1] “EPICS Home on WWW” http://www.aps.anl.gov/epics/

[2] N. Yamamoto *et al.*, Proceedings of the 7th international conference on accelerator and large experimental physics control systems, 4 - 8 October 1999,Trieste, Italy, pp.600-602.

[3] N. Yamamamoto *et al.*, Proceedings of the 7th European particle accelerator conference, Australian Academy of Sciences Press, Italy, 2000, pp. 1883-1885.

[4] A. Akiyama *et al.*, Contributed to IEEE Particle Accelerator Conference (PAC 99), New York, 29 Mar - 2 Apr 1999.

[5] T. Katoh *et al.*, “Present Status of the J-PARC Control System”, in these proceedings.

[6] George A. Howlett,“The BLT toolkit”, http://sourceforge.net/projects/blt/

[7] Greg McFarlane , “Pmw:Python Mega Widgets”, http://pmw.sourceforge.net/