DYNAMIC EXECUTION OF SCRIPTS ON EPICS IOC

N. Yamamoto, KEKB Control Group, KEK, Tsukuba, JAPAN
M. Takagi, Kanto Information Service, Tsuchiura, JAPAN

Abstract

Introduction of a scripting language on EPICS IOC(a VME single board computer) allows users to construct a flexible control system. Authors will discuss the possible applications of such scripting language and will show some of implementation of these applications.

1 INTERPRETED LANGUAGE IN A CONTROL SYSTEM

Use of the interpreted language in an accelerator control system is not a new idea. TRISTAN[1], predecessor of KEKB[2], control system used NODAL language developed at CERN[3] for its control system. Nodal language is a BASIC like interpreted language designed for a control system. One of its unique features is a remote execution of Nodal program within a program. As shown in a sample program, Figure1, a NODAL program can send a part of itself to the other computer and can get a result of execution in the remote computer.This function can be used to access resources only available at the specific computer. It can be also used to reduce data transfer overhead in some cases. This ability should be compared with the today's network agent technologies.

Experience of NODAL in the TRISTAN control system encouraged us to use interpreted languages in the KEKB control system. KEKB control system [4] was constructed upon the frame work of EPICS toolkit[5]. Scripting languages, SAD[7] and Python[8], are used in the system for the development of high level applications [6].

The followings are the few of merit for the use of interpreted languages in the control system.

1. User Participation These interpreted languages are easier to use than the compiler languages such as C, C++. End users , including accelerator physicist , developed many useful applications for the accelerator control.

2. incremental development Interpreted language can be also used for rapid prototyping of an application. Once specification is fixed, performance bottleneck part will be converted C/C++ modules. It reduces total efforts and time for the development.

3. Use of existing resources Usually these languages come with rich set of libraries, including GUI widgets. We can use these libraries where these are appropriate.

2 INTERPRETED LANGUAGE ON IOC

Although interpreted languages are successfully used in the KEKB control system, its use is limited only on a Unix workstations. These tools are also useful to test EPICS runtime database on IOC(Input Output Controller:VME single board computer with hardware interfaces) through CA(Channel Access:Data access protocol defined in EPICS toolkit). In this way, 1) you can access the data on IOC only through CA, and 2) it will increase network traffic when you need a large amount of data on IOC.

If we can use the interpreted language on IOC, 1) It can be used as diagnosis tools on IOC instead of VxShell in the vxWorks [1] . VxShell accept most of C expression but it cannot handle control statements such as if-statement. Interpreted language, such as Python, allows a user to write and to run a short program to test IOC, interactively. 2) It is also possible to send a scripts through the network and get the result of the script back from an IOC. It would automates a complex diagnosis procedure without increasing network traffic. 3) A script language can be used to extend behaviour of EPICS record. CALC record in the standard EPICS record set supports ".CALC" field where users can put a C-like expression. If user wants to test more complex expression which cannot fit in the limit of CALC record, user needs to use a SUBR(subroutine) record and a subroutine written in C. If EPICS can have CALC-like record which reads a Python script, it can be used instead of SUBR record/C function combination. PYTHON-record should allow user to change a script field dynamically as a CALC field in the CALC record. In this sense, Python-record will have a flexibility of CALC record and the power of SUBR record.

```
10.1 A=1; B= 2
10.2  EXEC <REMOTE> 20 A B C
        /*Execute a statement group 20
           on the CPU <REMOTE> */
20.1  C = A + B
20.2 REMIT C
       /*Return a SUM of A and B
          to the sender */
```

Figure 1: A sample NODAL program using remote execution of a program segment

[1] VxWorks is a registered trademark of Wind River Systems, Inc.

3 PYTHON ON VXWORKS

Python was firstly ported onto vxWorks by Jeff Stearns[2] . We have tested his port of Python and made modification we needed to compile and run it in our environment(HP-UX 10.2 as a host and Force Power Core 6750 as a target CPU). We also ported Python/EPICS CA(Channel Access) interface to the Python on VxWorks[9]. Python on Vxworks supports most of builtin functions(exit and quit are exception) and builtin modules. Some modules including a thread module, needs some work to run on VxWorks. The other modules may be irrelevant on VxWorks.

Python Interpreter uses static variables internally. It prevent us from having two copies of Python interpreter on same IOC. However, It will be still possible to run Python threads in a single Python interpreter, when thread module is supported in Python on VxWorks. For the first and second use of Python on IOC, alternative to the vxShell, it is not the restriction. For the third use of Python on IOC, extension tool of EPICS records, it is essential to support multiple instance of the interpreter or thread.

4 PYTHON SERVER ON EPICS IOC

In this section, we will see the sample sessions of remote execution of Python program on IOC. In these examples, we use python server program, pysvr.c, which is included in the standard Python distribution. When pysvr is running it waiting for a connection from client. Once a connection is established, it read Python statements sent by the client , execute it in its interpreter, and then returns the result as a string to the client. The client can be any program which can send and receive data to/from TCP socket, telnet for example.

The figures 2 and 3 show a sample session of pysvr. In this example, pysvr program runs on IOC and a clients is a telnet program running on HP-UX workstation.

In the second example(Figures 4 and 5), a python program on a host workstation uses telnetlib module to establish connection to a pysvr on an IOC and send python script to the remote IOC for execution. Results of each execution will be printed out on the terminal on the workstation. In the program sent to the IOC, an average of data in a waveform record with 1024 elements is calculated. Calculation of average is performed on IOC, so it is not necessary to send all values in the waveform record through a network. It reduces the potential traffic on the network considerably.

As shown in these examples, the pysvr.c program proves the possibility and usefulness of remote execution of Python scripts on IOC. However, more improvements are required for the use in the control system. Firstly, Python on VxWorks should support a thread module so that pysvr can accept multiple connections. Threads can be easily mapped to tasks on vxWorks, and vxWorks support semaphore library. It should not be difficult to develop wrapper codes for Python interpreter on VxWorks.

[2]see http://www.python.org/download/download_other.html for more information

```
Connected to IOCEPAC.
Escape character is '^]'.
>import ca
>ch=ca.channel("CO_IOC:COCCC:CLOCK")
>ch.wait_conn()
>ch.get();ch.pend_event(0.1)
10
>ch.val
'2000/06/19 14:25:20'
>import sys
>sys.exit()
```

Figure 2: Sample log of a PYSVR session on host computer side

```
-> taskSpawn "PythonServer",184,0,128000,pysvr
Listening on port 4000...
Start thread for connection 18.
run_command: import ca
run_command: ch=ca.channel("CO_IOC:COCCC:CLOCK")
run_command: ch.wait_conn()
run_command: ch.get();ch.pend_event(0.1)
run_command: ch.val
run_command: import sys
run_command: sys.exit()
End thread for connection 18.
```

Figure 3: Sample log of a PYSVR session on IOC side

Secondly, it should returns a value as a binary format rather than as a string , which current version of pysvr program returns. If a pysvr client program get result as a string, it should be evaluated again before it is used in the client program. Use of binary data format can reduce overhead of conversion of a return value to and from string form. Python already has a module to serialise structured data and object, Pickle and cPickle, in the standard Python library . Original data can be easily retrieved from pickled data using the loads() function. These library can be used as a base of binary data exchange over network.

5 PYTHON ON IOC AS AN EXTENSION TOOL OF EPICS RECORD

Once Python on vxWorks supports the thread module, new EPICS record type PYTHON, which evaluate Python scripts in its process function, can be developed. Global lock for Python interpreter is created at IOC initialization and Python thread is created at record initialization.A script attached to the record will be compiled into an intermediate code using Python's compile function and this intermediate code will be evaluated at the record processing. API to access EPICS runtime database can be easily developed using SWIG(Simple Wrapper Interface Generator)[10].

```
import telnetlib, time, ca
host=("iocepac",4000)
def test():
    try:
        tn=apply(telnetlib.Telnet,host)
    except telnetlib.socket.error:
        print "cannot connect to ",host
        return None
    except:
        print "telnet open error"
        return None
    res=tn.read_until(">")
    print res
    def execute(line,tn=tn):
        tn.write(line+"\n")
        res=tn.read_until(">")
        print "%s"%line
        print res
    for line in ("import ca",
"ch=ca.channel(\"sample:wf:double\")",
"ch.pend_event(0.1)",
"ch.get(); ch.pend_event(0.01)",
"def add(x,y): return x+y",
"ave=reduce(add, ch.val)/len(ch.val);ave",
):
        execute(line)
    tn.close()
if __name__ == "__main__":
    test()
```

Figure 4: a sample python program which runs on a host computer

```
>>> test()
>
import ca
>
ch=ca.channel("sample:wf:double")
>
ch.pend_event(0.1)
10
>
ch.get(); ch.pend_event(0.01)
10
>
def add(x,y): return x+y
>
ave=reduce(add, ch.val)/len(ch.val);ave
511.5
```

Figure 5: output of the python program shows in Figure 4

PYTHON record can be used where CALC record is used and it can be used to prototype a subroutine used with SUBROUTINE record. Dynamical reloading of library and scripts makes development of a prototype easy.

6 CONCLUSION

We pointed out the possible use of interpreted language on an EPICS IOC. Prototype examples of python server on IOC were shown. Problems of current implementation were pointed out and outline of solutions is discussed.

REFERENCES

[1] S-I Kurokawa, et al., “The TRISTAN Control System”, Nucl. Instr. and Meth., A247, (1986) pp. 29-36. ; T. Mimashi et al., “The rejuvenation status of TRISTAN accelerator control system”, Nucl. Instr. and Meth., A352 (1994) 128-130.

[2] “KEKB B-Factory Design Report”, KEK Report 95-7, August 1995

[3] M.C. Crowley-Milling and G.C. Shering, “The NODAL System at the SPS”, CERN 78-08

[4] T. Katoh et al., “Present Status of the KEKB Control System”, ICALEPCS ’97, Beijing, China, November 3-7, 1997; N. Yamamoto et al., “KEKB control system: the present and the future.”, PAC 99, New York City, March 1999.

[5] L. Dalesio et al. , “The Experimental Physics and Industrial Control System Architecture: Past, Present, and Future”, Proc. ICALEPCS, Berlin, Germany, 1993, pp 179-184. W. McDowell et al.:EPICS Home Page, http://epics.aps.anl.gov/asd/controls/epics/EpicsDocumentation/ EpicsGeneral/

[6] N. Yamamoto et al, “USE OF OBJECT ORIENTED INTERPRETIVE LANGUAGES IN AN ACCELERATOR CONTROL SYSTEM”, ICALEPCS 99, Trieste, Italy, 1999

[7] K.Oide et al, “SAD home page”, URL:http://www-acc-theory.kek.jp/SAD/sad.html

[8] M. Lutz, “Programming Python”, O’Reilly & Associates, Inc. USA, 1996; G.van Rossum, “Python Home Page”, URL: http://www.python.org/

[9] N. Yamamoto, ”Python/Tk in EPICS”, EPICS collaboration meeting, KEK, May, 2000,URL:http://www-acc.kek.jp/EPICS_meeting/Presentations/NoboruYamamoto/PythonTK.html

[10] D.M. Beazley, “SWIG:Simplified Wrapper and Interface Generator”, URL: http://www.swig.org/