SERVER DEVELOPMENT FOR NSLS-II PHYSICS APPLICATIONS AND PERFORMANCE ANALYSIS*

Guobao Shen #, Marty Kraimer, BNL, Upton, NY 11973, U.S.A.

Abstract

The beam commissioning software framework of NSLS-II project adopts a client/server based architecture to replace the more traditional monolithic high level application approach. The server software under development is available via an open source sourceforge project named epics-pvdata, which consists of modules pvData, pvAccess, pvIOC, and pvService. Examples of two services that already exist in the pvService module are itemFinder, and gather. Each service uses pvData to store in-memory transient data, pvService to transfer data over the network, and pvIOC as the service engine. The performance benchmarking for pvAccess and both gather service and item finder service are presented in this paper. The performance comparison between pvAccess and Channel Access are presented also.

INTRODUCTION

For an ultra low emittance synchrotron radiation light source like NSLS II, the control system requirements, especially for beam control are tight. To control and manipulate the beam effectively, a use case study has been performed to satisfy the requirement [1] and theoretical evaluation has been performed [2]. The analysis shows that model based control is indispensable for beam commissioning and routine operation. However, there are many challenges such as how to re-use a design model for on-line model based control, and how to combine the numerical methods for modeling of a realistic lattice with the analytical techniques for analysis of its properties.

To satisfy the requirements and challenges, adequate system architecture for the software framework for beam commissioning and operation is critical. The existing traditional approaches are self-consistent, and monolithic. Some of them have adopted a concept of middle layer to separate low level hardware processing from numerical algorithm computing, physics modelling, data manipulating and plotting, and error handling. However, none of the existing approaches can satisfy the requirement. A new design has been proposed by introducing service oriented architecture technology [3][4][5][6][7], and client interface is undergoing [8].

The design and implementation adopted a new EPICS implementation, namely epics-pvdata [9], which is under active development. The implementation of this project under Java is close to stable, and binding to other language such as C++ and/or Python is undergoing.

* Work supported under auspices of the U.S. Department of Energy under Contract No. DE-AC02-98CH10886 with Brookhaven Science Associates, LLC, and in part by the DOE Contract DE-AC02-76SF00515

#shengb@bnl.gov

In this paper, we focus on the performance benchmarking and comparison for pvAccess and Channel Access, the performance evaluation for 2 services, gather and item finder respectively.

PVACCESS PERFORMANCE

PvAccess is a new communication protocol that supports the transfer of arbitrary data structures rather than the fixed set of data types supported by Channel Access, and will act as a successor of Channel Access.

To satisfy the requirements as described in [1][2], a performance benchmarking is indispensable. A study was performed to measure and compare the performance of Channel Access and pvAccess.

Environment Configuration

The system configuration is as shown in Figure 1. All servers and client are running Debian LINUX system.

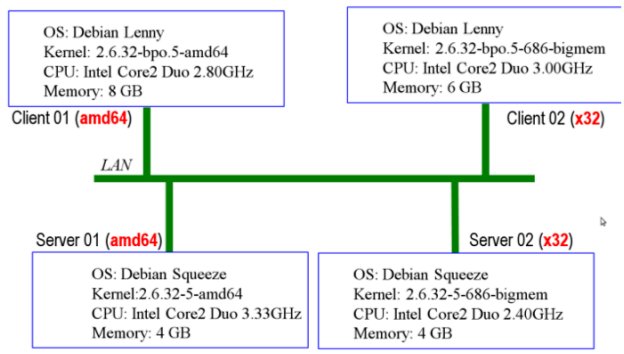

Figure 1: Benchmarking Environment Configuration.

The router used in this benchmarking was a GS108 GIGABIT switch from NETGEAR, which has 8 ports. All clients and servers are connected to this router through a wired network.

Four (4) different cases were performed by combining the client and server in different ways, as listed in Table 1.

Table 1: Client-Server Combinations

Test Case	Client	Server
1	Client 01	Server 01
2	Client 02	Server 02
3	Client 01	Server 01
4	Client 02	Server 02

Array Processing Performance

The purpose of this benchmarking is to measure the array transfer rate, with varying array sizes, between array records on a pvIOC and a client. A client sends out a

command to get the value from 10 array records. When all arrays are received, the next command is sent. For this benchmarking, pvAccess sends network packets without delay. For small arrays pvAccess has better performance and for large arrays the performance is identical. Figure 2 shows the results.

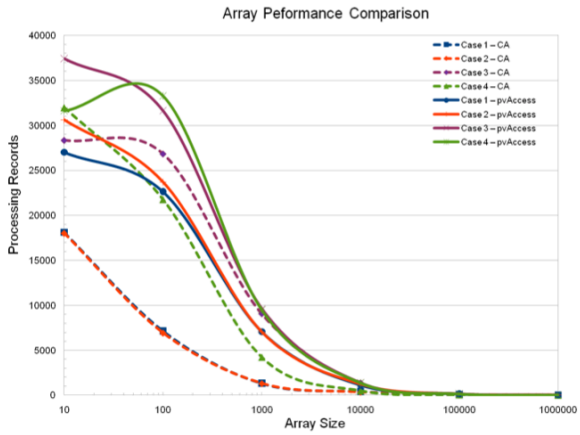

Figure 2: Array performance and comparison between CA and pvAccess. The horizontal axis is record array size. Vertical axis is records processed per second (also the number of array transferred per second). Dashed lines are for CA, and solid lines are for pvAccess.

ProcessGet Performance

This test compares the performance of a scalar database under different client/server combinations. The client issues a processGet (process and get result) request to a number of channels. A plot to show how many records can be processed in one second is shown as Figure 3.

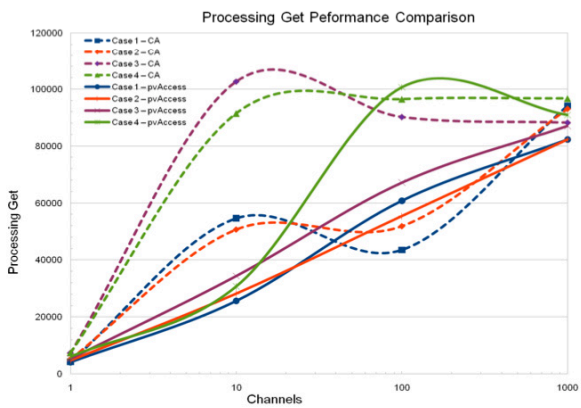

Figure 3: ProcessGet performance and comparison between CA and pvAccess. The horizontal axis is number of channels (log scale). Vertical axis is processGet per second. Dashed lines are for CA, and solid lines are for pvAccess.

This test indicates that for simultaneous processGets for many channels, pvAccess and CA have similar performance, but for few channels CA is better.

A full report can be found in [11], which describes the performance of pvAccess and compares it with that of Channel Access.

SERVICE PERFORMANCE

There are several services have been prototyped as described in [6][7]. The performance benchmarking was conducted for 2 major services, item finder service and gather service respectively. Original implementation for above 2 services can be found in [6], and also the data flow. More detail can be found in [7].

Gather Performance

Basic idea of the gather service is that a client sends a PV list with a string to this service; the service then creates a pvRecord dynamically with the record name given by the client. The new created pvRecord acts as a bridge between low level V3 IOCs, and the client. The gather service supports put, get and monitor operation.

The performance was evaluated under the environment shown in Figure 1, and client server combinations are shown in Table 2.

Table 2: Client-Server Combinations for Gather

Test Case	Client	Server	V3 IOC
1	Server 01	Server 01	Server 01
2	Client 02	Server 01	Server 01
3	Client 02	Server 01	Server 02

The benchmarking was performed for get and put operation with scalar channel. The result shows that to process 1,000 channels, the time is less than 10 ms. In another word, it has a capability to update 1,000 PVs with a frequency of 100Hz. The result for a varying number of V3 channels is shown as Figure 4.

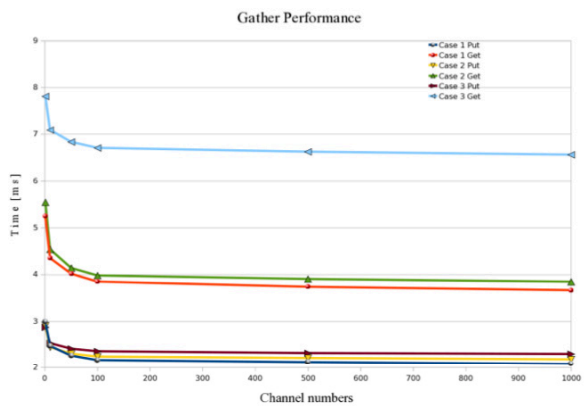

Figure 4: Gather service performance for get and put processing with different number of V3 channels. The horizontal axis is number of V3 channel, and vertical axis is processing time in milliseconds.

Item Finder Performance

The item finder service provides a dictionary service. The basic idea is to get a list of physics elements and its

associated properties such as EPICS PV names for read-back, set-point, temperature, and so on. It is designed and prototyped against MySQL relational database (RDB). A use case is to query all horizontal BPM PV names with specified physics name.

The performance was evaluated under the environment shown in Figure 1, and client server combinations shown in Table 3. The underneath relational database, which uses MySQL as the RDBMS, runs on server 01.

Table 3: Client-Server Combinations for Item Finder

Test Case	Client	Server
1	Client 02	Server 01
2	Server 01	Server 01

The result is demonstrated as Figure 5. A query for 1,400 items takes less than 30 ms.

Figure 5: Item finder performance. The horizontal axis is item number to be queried, and vertical axis is processing time with unit in milliseconds.

SUMMARY

In this paper, we conducted a performance benchmarking for pvAccess protocol, and compared the performance of pvAccess with that of Channel Access. The results shows that pvAccess has a better performance when processing array. For processGet operation, pvAccess has identical performance for many hannels, but a slight worse performance for few channels. The benchmarking was conducted using pvAccess implemented in Java, and Channel Access implemented in C/C++. A future benchmarking will be carried out once a C++ version of pvAccess is available.

The gather service and item finder service demonstrated a reasonable performance, and which is acceptable for physics application development.

ACKNOWLEDGEMENT

The authors would like to thank Matej Sekoranja at COSYLAB for his contributions on epics-pvdata development. They want to give their thanks to Leo Dalesio at BNL for his continuous support and encouragement.

REFERENCES

[1] J. Bengtsson, *et al*, "NSLS-II: Model Based Control – A Use Case Approach", NSLS-II Tech Note 51 (2008).

[2] J. Bengtsson, "Design and Control of Ultra Low Emittance Light Sources", Proc. of ICAP09 (2009), TU3IOPK04, San Francisco, USA.

[3] G. Shen, "A Software Architecture for High Level Applications", Proc. of PAC09 (2009), FR5REP004, Vancouver Canada.

[4] G. Shen, "A Modular Environment for High Level Applications", Proc. of ICALEPCS09 (2009), THP094, Kobe Japan.

[5] P. Chu, *et al*, "Service Oriented Architecture for High Level Applications", Proc. of IPAC10 (2010), TUPEC072, Kyoto Japan.

[6] G. Shen, *et al*, "Prototype of Beam Commissioning Environment and its Applications for NSLS-II", Proc. of IPAC10 (2010), WEPEB026, Kyoto Japan.

[7] G. Shen, *et al*, "A Novel Approach for Beam Commissioning Software using Service Oriented Architecture", Proc. of PCaPAC10 (2010), WEPL037, Saskatoon Canada.

[8] G. Shen, *et al*, "NSLS-II High Level Application Infrastructure and Client API Design", this Proc., MOP250.

[9] http://sourceforge.net/projects/epics-pvdata/

[10] M. R. Kraimer, *et al*, "Evolution of the EPICS Channel Access Protocol", Proc. of ICALEPCS09 (2009), MOD005, Kobe Japan.

[11] G. Shen, "Performance Analysis of EPICS Channel Access and pvAccess", NSLS-II Tech Note 082 (2010).

[12] G. Shen, *et al*, "Services Development for NSLS-II Physics Application Environment using pvService", EPICS Collaboration Meeting Fall 2010, BNL.