NSLS-II TECHNICAL NOTE BROOKHAVEN NATIONAL LABORATORY	NUMBER **082**
AUTHOR **GUOBAO SHEN**	DATE **10/29/2010**
TITLE **Performance Analysis of EPICS Channel Access and pvAccess**	

Abstract

This note provides measurement results on the performance of Channel Access and pvAccess. Channel Access is the communication layer of the EPICS software framework used by the NSLS-II project, and 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. The beam commissioning environment of NSLS-II has adopted the Client/Server model, and will use pvAccess as its data communication bus. This report describes the performance of pvAccess and compares it with that of Channel Access.

For this benchmarking we used 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 released.

I. OVERVIEW

This note provides measurement results on the performance of Channel Access (CA) and pvAccess. Channel Access is the communication layer of the EPICS software framework used by the NSLS-II project. PVAccess is a new communication protocol that supports the transfer of arbitrary data structures rather than the fixed set of data types supported by CA. The beam commissioning environment of NSLS-II has adopted a Client/Server model, and will use pvAccess as its data communication bus. This report describes the performance of pvAccess and compares it to that of Channel Access.

II. SYSTEM ENVIRONMENT

1. System Configuration

The pvAccess used in this benchmarking is implemented in Java, and the version for CA/EPICS is EPICS released R3.14.11. The language combination is as shown in Table 1.

Table 1: Client-Server Language Binding.

Test Case	Client	Server
pvAccess	Java	Java
CA	C/C++	C/C++

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

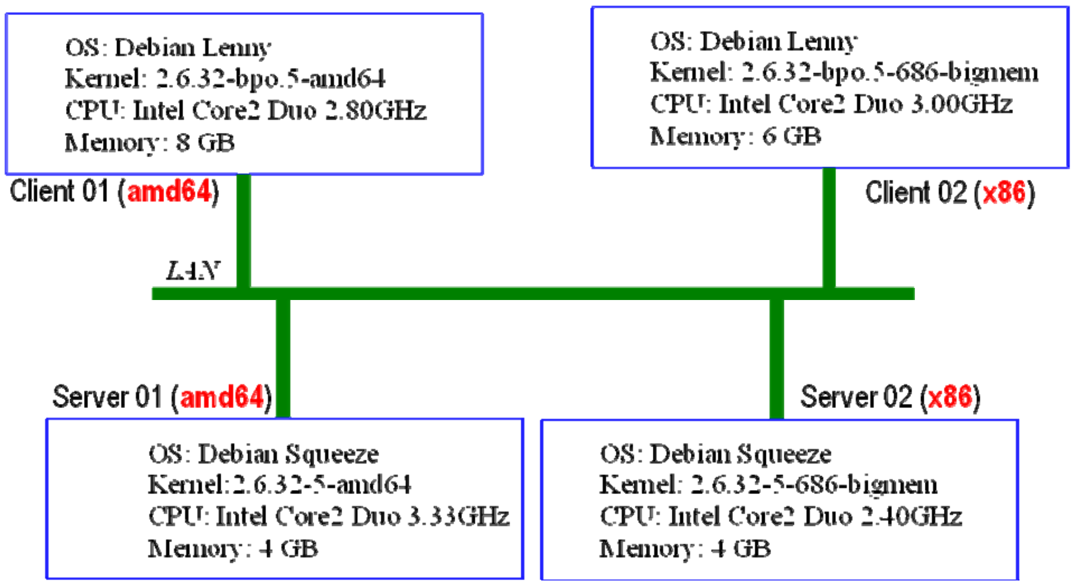

Figure 1: Machine configuration and network connection.

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

2. Runtime Database Preparation

Two databases are provided for both CA and pvAccess: one for scalar data and one for array data. The scalar database consists of 1,000 counter records named from counter00000 to counter00999. When a counter record is processed and the value is <999, the value is incremented by 1. When the value is 999, it is set equal to 0. Alarms are set as follows:

Table 2: Alarm Setting for Record.

LOLO	LOW	HIGH	HIHI
5	10	990	995

The array database consists of 60 array records. Each array record is an array of doubles. There are 10 records for each different array size, and the array sizes are 10, 100, 1,000, 10,000, 100,000 and 1,000,000, respectively. The array is initialized to 0, 1, …, to the array length (999, for example, if the array size is 1,000). Each array element is increased by 1.0 if an array record is processed.

3. Test Case

The following tests are performed:

Array. This has a database with 10 array records. Each array is doubles, which has 10 to 1000000 elements as described above. A processGet is issued for each array. The processGet is an atomic operation in pvAccess. For CA, since it does not support processGet, it issues a command to force processing each counter record and then issues a get to each of the records. The performance test measures the number of processGets per second for each different array size.

processGet. This has a database with 1,000 passive counter records. The performance test repeatedly issues a processGet to each of the counter records.

Event. The database has 1,000 counter records, each scanned on the same event and a single event record which is a passive record. The test repeatedly asks the event record to process and then waits until it receives a monitor from the 1,000th counter record.

III. RESULTS

Four (4) different test cases are performed by combining the client and server in different ways, as listed in Table 3.

Table 3: Client-Server Combination.

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

In current pvAccess implementation, the waiting time to send out a TCP queue is configurable. The user has the flexibility to choose sending message by message or wanting for a while. In this test, four different waiting times are benchmarked, as below:

0; (no delay)

1 ms

5 ms

10 ms

1. Array Performance

The purpose of this measurement is to measure how many array records a client can get value back from a server when giving a record a different array size. In a different client/server combination, a client sends out a command to get value from 10 array records. Once the client gets all value back from the server, it sends out the next command. Figure 2 shows how many records can be processed in one second. The plot is in five groups from left to right, for CA and pvAccess with different waiting times (0, 1, 5, 10), respectively.

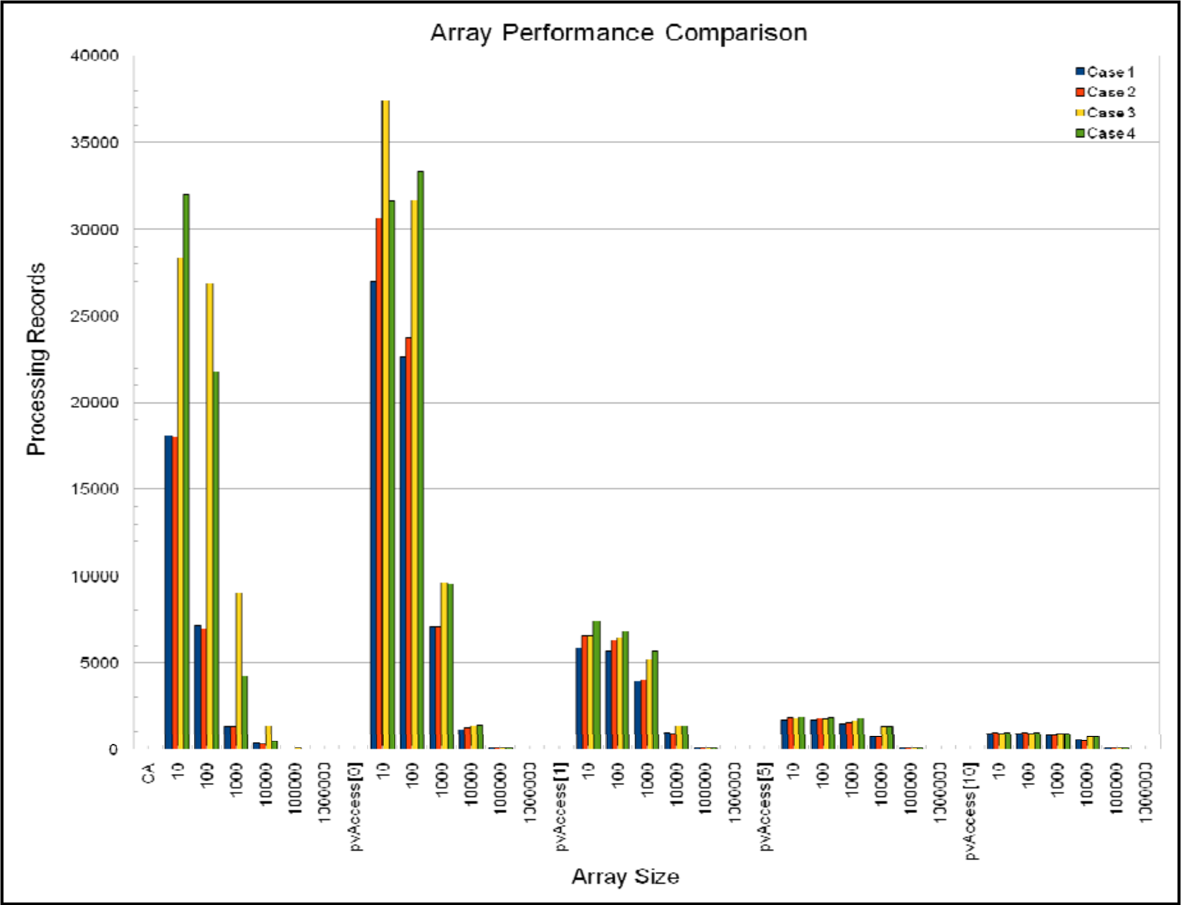

Figure 2: Array Performance. The horizontal axis is different array size for CA and pvAccess with different waiting time: pvAccess [0] means transporting TCP message by message; pvAccess [1] waiting for 1 ms; pvAccess [5] for 5 ms; and pvAccess [10] for 10 ms. Vertical axis is records number and can be processed within 1 second.

From Figure 2, one can see that the performance of CA and pvAccess is comparable when pvAccess sends out a TCP package message by message. The pvAccess performance decreases rapidly when waiting longer.

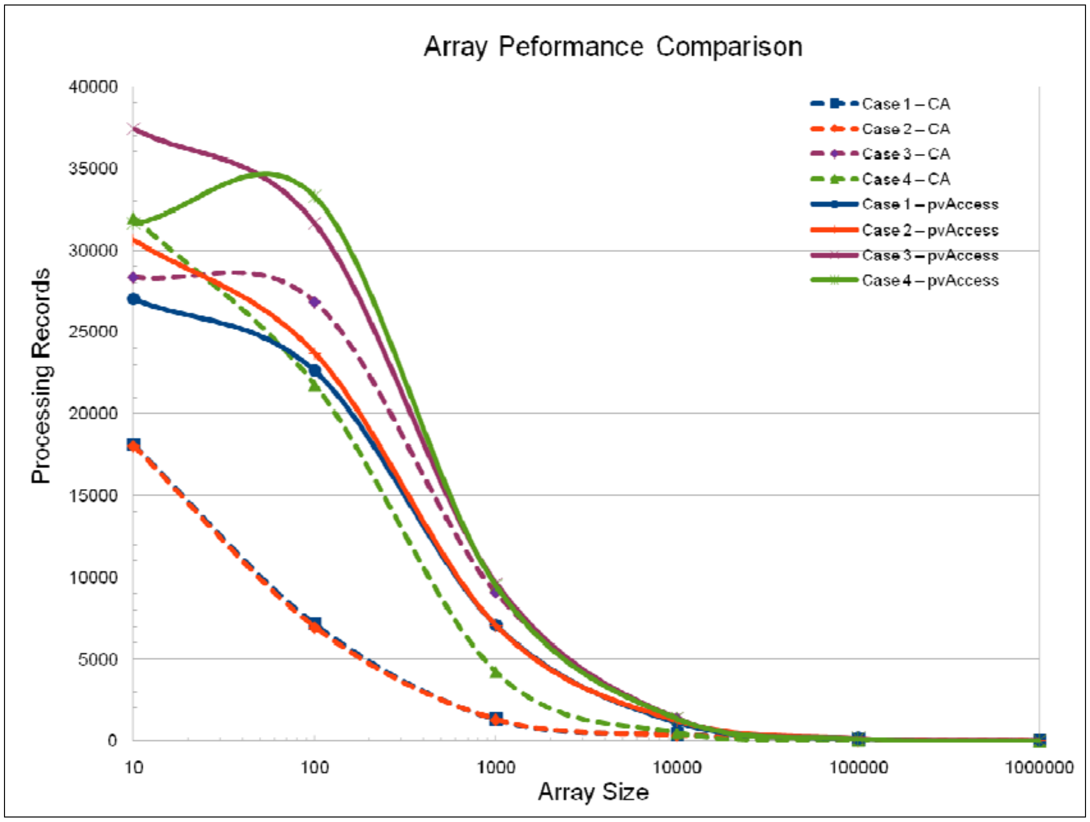

Figure 3: Array Performance Comparison between CA and pvAccess with waiting time = 0. The horizontal axis is record array size for CA and pvAccess. Vertical axis is records number can be processed within one second. The dashed line is for CA, and the solid line is for pvAccess.

Figure 3 shows a detailed comparison between CA and pvAccess when pvAccess sends out a TCP package without blocking any message. When array size is small, pvAccess has better performance, and when the array size is large, the performance is identical. Detailed results can be found in Appendix I.

To compare the impact caused by CPU architecture (x32/x64), we compared the processing results and CA performance, as shown in Table 4.

Table 4: CA Performance Comparison (x86/amd64).

ARRAY	Case 2/1	Case 4/3	Case 2/4	Case 1/3
10	0.99	1.13	0.56	0.64
100	0.97	0.81	0.32	0.27
1,000	0.99	0.47	0.31	0.14
10,000	0.83	0.37	0.67	0.30
100,000	0.76	0.30	0.59	0.23
1,000,000	1.00	0.30	0.75	0.22

- Case 2/1: same server (x32), x32/x64 client. Same IOC server runs on Server 02, which has a 32-bit processor. The performance from a 32-bit client and 64-bit client is almost the same, and array size has less impact on the performance. It means the performance is limited by the 32-bit client.

- Case 4/3: same server (x64), x32/x64 client. Same IOC server runs on Server 01, which has a 64-bit processor. The performance from a 32-bit client and 64-bit client shows significant difference. The array size has big impact on the performance; also, the performance decreases when array size increases. It means the performance is limited by the 32-bit client.
- Case 2/4: same client (x32), x32/x64 server. Client runs on Client 02, which has a 32-bit processor. The client's performance with a 32-bit server and 64-bit server shows a large difference, but the array size has less impact on performance. It means the performance is limited by both 32-bit client servers.
- Case 1/3: same client (x64), x32/x64 server. Client runs on Client 02, which has a 64-bit processor. The client's performance t from a 32-bit server and 64-bit server shows a large difference. The array size has great impact on performance.

For this comparison, we did not count the impact caused by the CPU clock, network capability, and memory capacity. A similar analysis has been done for pvAccess, and the results are shown in Table 5.

Table 5: pvAccess Performance Comparison (x86/amd64).

	CASE															
	2/1	4/3	2/4	1/3	2/1	4/3	2/4	1/3	2/1	4/3	2/4	1/3	2/1	4/3	2/4	1/3
ARRAY	Delay = 0				Delay = 1				Delay = 5				Delay = 10			
10	1.13	0.84	0.97	0.72	1.12	1.13	0.89	0.90	1.06	1.05	0.96	0.96	1.05	1.04	0.98	0.98
100	1.05	1.05	0.71	0.72	1.12	1.06	0.92	0.88	1.06	1.04	0.98	0.96	1.04	1.03	0.98	0.98
1,000	1.00	0.99	0.74	0.73	1.02	1.09	0.71	0.76	1.03	1.07	0.88	0.91	1.02	1.04	0.93	0.95
10,000	1.09	1.00	0.89	0.82	0.95	1.01	0.66	0.71	1.00	1.00	0.61	0.61	0.98	1.01	0.67	0.69
100,000	1.01	1.00	1.00	0.99	1.00	1.00	1.01	1.00	1.00	1.00	0.99	1.00	1.00	1.00	0.99	0.99
1,000,000	0.99	0.99	0.97	0.97	0.99	1.00	0.96	0.98	1.01	0.97	1.01	0.97	1.00	1.01	0.99	1.01

Tables 2 and 3 indicate that array size has less impact to pvAccess performance on a 32-bit CPU or a 64-bit CPU.

2. Processing Get Performance

The processing get compares the performance of the scalar database under different client/server combinations, when the client issues a processGet request. A client monitors different records, 1, 10, 100, and 1,000, at the same time. For CA, processGet means a client forces an IOC to process a record, and then get the result back. Note that all numbers are thousands of processGets per second. A plot to show how many records can be processed in one second is shown as Figure 4. The plot is in five groups from left to right, for CA and pvAccess with different waiting times (0, 1, 5, 10), respectively.

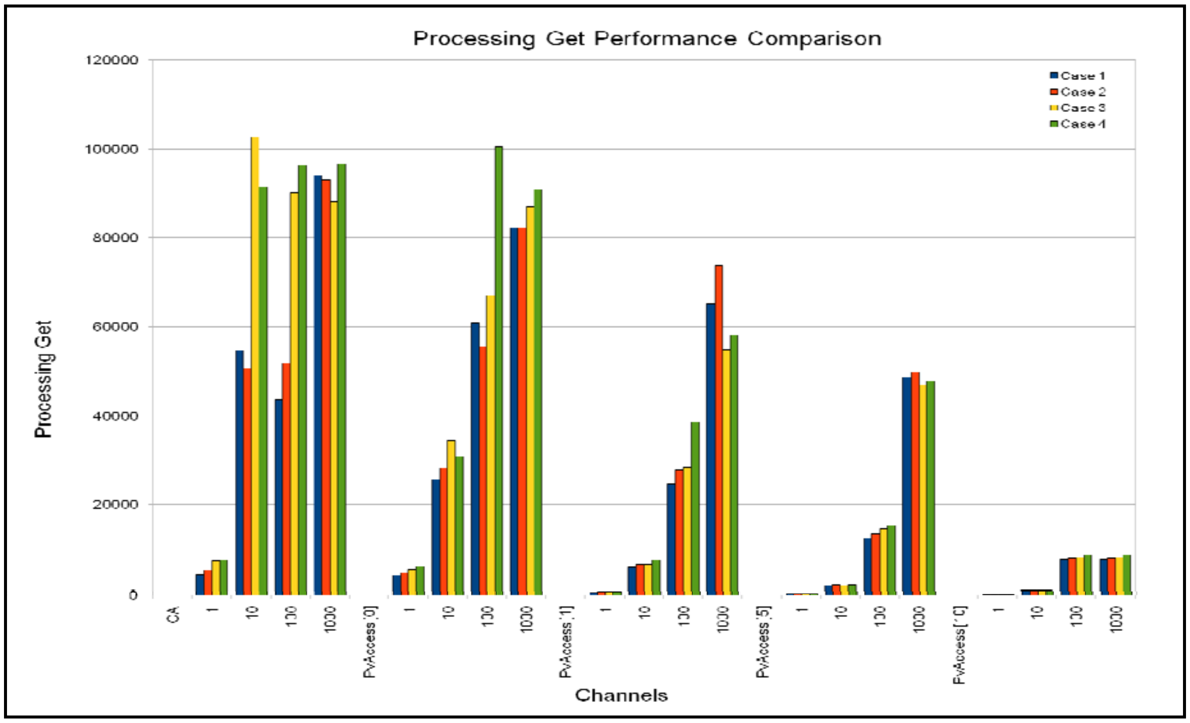

Figure 4: Process Get Performance. The horizontal axis is different channels monitored by client for CA and pvAccess with different waiting time: pvAccess [0] means transporting TCP message by message; pvAccess [1] waiting for 1 ms; pvAccess [5] for 5ms; and pvAccess [10] for 10 ms. Vertical axis is records number can be processed within one second.

Figure 4 indicates that the best performance gained for pvAccess is when it transports a TCP package message by message. The pvAccess performance decreases rapidly when waiting longer. Even in that case, CA has better performance than pvAccess, especially when a client monitors 10 records at the same time. A detailed comparison can be found in Figure 5, which compares the performance between CA and pvAccess with transporting a TCP package message by message. Detailed results can be found in Appendix II.

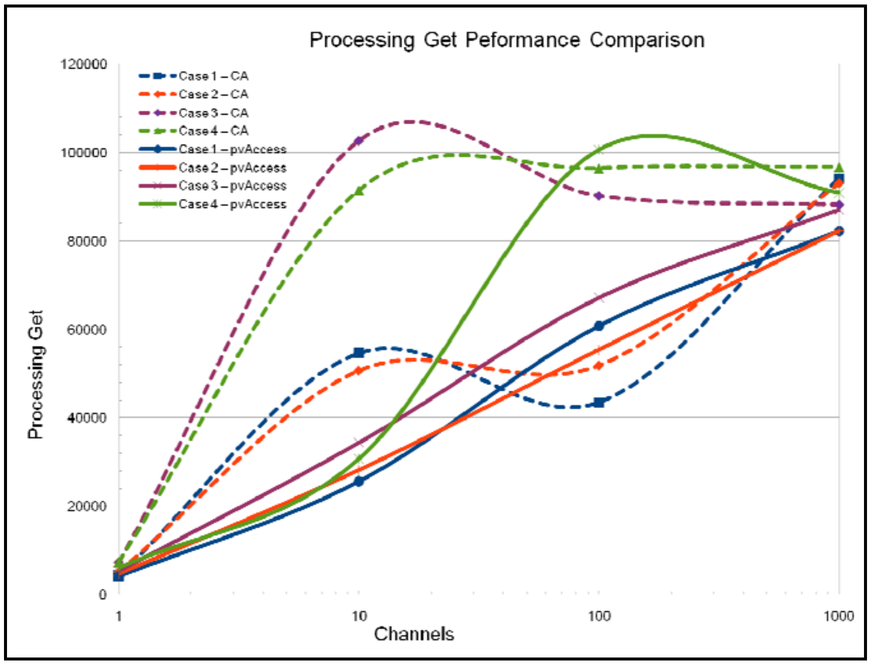

Figure 5: Processing Get Performance Comparison between CA and pvAccess without blocking TCP message transport. The horizontal axis is record array size for CA and pvAccess. Vertical axis is records number can be processed within 1 second. The dashed line is for CA, and the solid line is for pvAccess.

To compare the impact caused by CPU architecture (x32/x64), we compared the processing results and CA performance, as shown in Table 6.

Table 6: CA Performance Comparison (x86/amd64).

	CASE			
CHANNEL	2/1	4/3	2/4	1/3
1	1.22	1.02	0.70	0.59
10	0.93	0.89	0.55	0.53
100	1.19	1.07	0.54	0.48
1,000	0.99	1.10	0.96	1.07

- Case 2/1: same server (x32), x32/x64 client. Same IOC server runs on Server 02, which has a 32-bit processor. The performance from a 32-bit client and 64-bit client is almost the same.
- Case 4/3: same server (x64), x32/x64 client. Same IOC server runs on Server 01, which has a 64-bit processor. The performance from a 32-bit client and 64-bit client is almost the same.
- Case 2/4: same client (x32), x32/x64 server. Client runs on Client 02, which has a 32-bit processor. The performance got on client from a 32-bit server and 64-bit server has a big difference, but the number of monitored channels has less impact on performance.
- Case 1/3: same client (x64), x32/x64 server. Client runs on Client 02, which has a 64-bit processor. The performance got on client from a 32-bit server and 64-bit server is almost the same when the number of monitored channels is less than 100. When monitoring 1,000 channels, the performance is almost the same.

For this comparison, we did not count the impact caused by the CPU clock, network capability, and memory capacity. A similar analysis has been done for pvAccess, and the results are shown in Table 7. Tables 2 and 3 show the array size has less impact to pvAccess performance on a 32-bit CPU or a 64-bit CPU.

Table 7: pvAccess Performance Comparison (x86/amd64).

	CASE															
	2/1	4/3	2/4	1/3	2/1	4/3	2/4	1/3	2/1	4/3	2/4	1/3	2/1	4/3	2/4	1/3
CHANNEL	Delay = 0				Delay = 1				Delay = 5				Delay = 10			
1	1.14	1.14	0.77	0.77	1.11	1.10	0.92	0.91	1.06	1.05	0.98	0.97	1.04	1.04	0.99	0.98
10	1.10	0.89	0.92	0.75	1.11	1.15	0.88	0.91	1.04	1.05	0.97	0.97	1.03	1.04	0.98	0.98
100	0.91	1.50	0.55	0.91	1.13	1.36	0.72	0.87	1.08	1.05	0.87	0.85	1.04	1.09	0.92	0.96
1000	1.00	1.04	0.91	0.95	1.13	1.06	1.27	1.19	1.02	1.02	1.04	1.04	1.04	1.09	0.92	0.96

3. Processing Event Performance

The processing event compares the performance of the scalar database under different client/server combinations with client monitoring event requests. A client monitors different records, 1, 10, 100, and 1,000, respectively, at the same time. For CA, process event means a client forces an IOC to process an event record to generate an event. The counting records will be trigged by that event and get processed. The client gets the result back and sends the next processing request. Note that all numbers are “thousands processed per second.” Figure 6 shows how many records can be processed in 1 second. The plot is in five groups from left to right; these are for CA and pvAccess with different waiting times (0, 1, 5, 10), respectively. Plots a and b are in different axis scales: normal scale for plot (a) and log10 scale for plot (b).

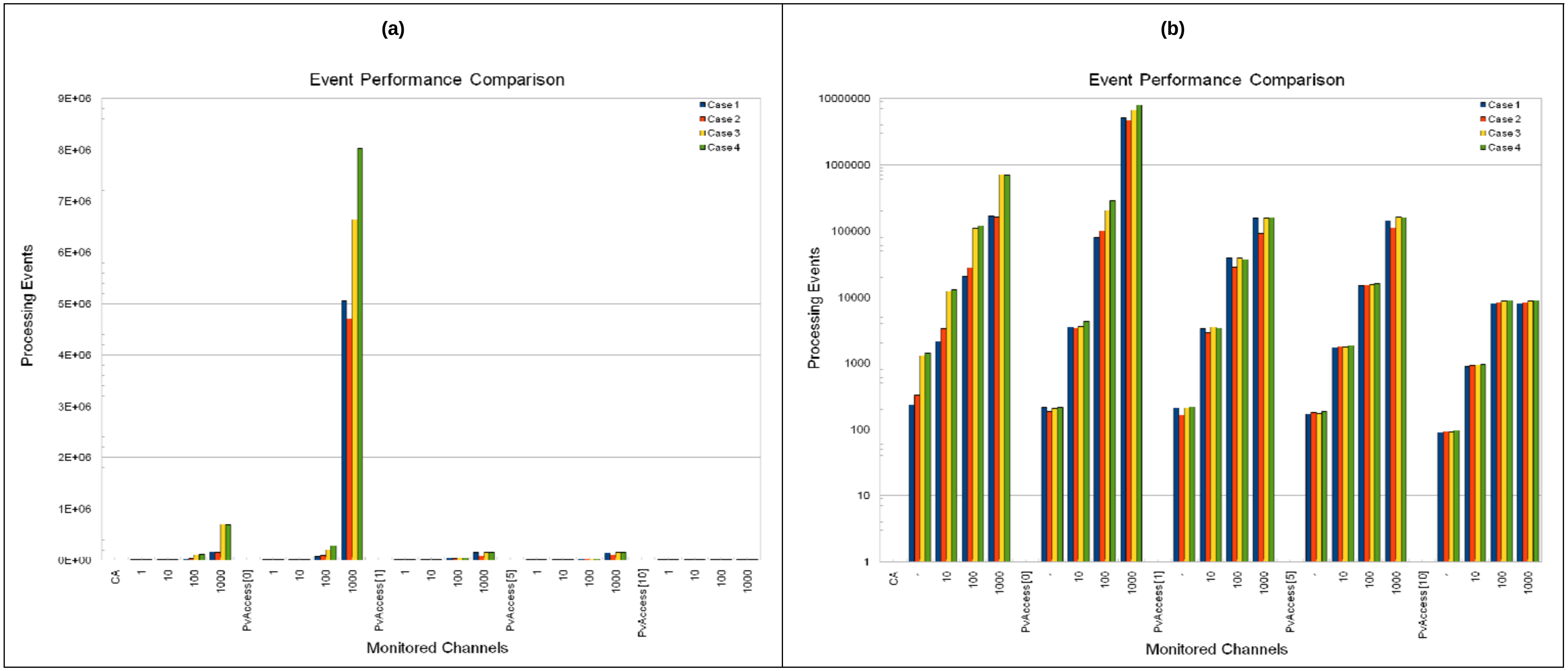

Figure 6: Event Processing Performance. The horizontal axis is different channels monitored by CA and pvAccess with different waiting time: pvAccess [0] means transporting TCP message by message; pvAccess [1] waiting for 1 ms; pvAccess [5], waiting for 5 ms; and pvAccess [10], waiting for 10 ms. Vertical axis is the number of records that can be processed within 1 second. The vertical axis of plot (b) is in log10-scale, with normal scale for plot (a).

From Figure 6, it can be seen that the performance of pvAccess is best when pvAccess transports a TCP package message by message. The pvAccess performance decreases rapidly when waiting longer.

Figure 7 shows a detailed comparison between CA and pvAccess when pvAccess sends out a TCP package without blocking any messages. When array size is small, CA pvAccess has similar performance, and when the array size is large, the pvAccess is significantly better. Detailed results can be found in Appendix III.

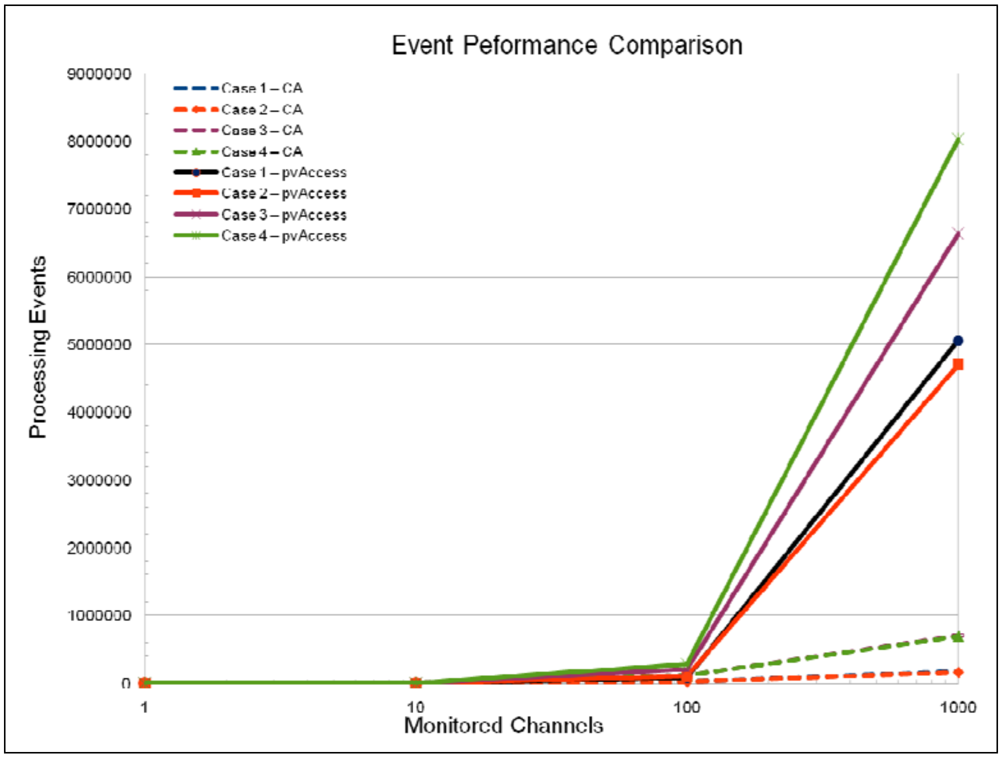

Figure 7: Event Processing Performance Comparison between CA and pvAccess with waiting time = 0. The horizontal axis is channels monitored by CA and pvAccess. Vertical axis is records number can be processed within 1 second.

To compare the impact caused by CPU architecture (x32/x64), we compared the processing results and CA performance, as shown in Table 8.

Table 8: CA Performance Comparison (x86/amd64).

	CASE			
CHANNEL	2/1	4/3	2/4	1/3
1	1.40	1.10	0.23	0.18
10	1.57	1.07	0.25	0.17
100	1.34	1.08	0.23	0.19
1000	0.97	0.98	0.23	0.24

- Case 2/1: same server (x32), x32/x64 client. The same IOC server runs on Server 02, which has a 32-bit processor. The performance from a 32-bit client is better than from a 64-bit client when monitored channels are fewer than 100, and almost the same when 1,000 channels are being monitored.
- Case 4/3: same server (x64), x32/x64 client. The same IOC server runs on Server 01, which has a 64-bit processor. Performance from a 32-bit and 64-bit client is almost the same, and the number of monitored channels shows slight impact on performance.
- Case 2/4: same client (x32), x32/x64 server. Client runs on Client 02, which has a 32-bit processor. The performance got on client from a 64-bit server is much better than from a 32-bit server, but the number of monitored channels has less impact to the performance.
- Case 1/3: same client (x64), x32/x64 server. Client runs on Client 02, which has a 64-bit processor. The performance got on client from a 64-bit server is much better than from a 32-bit server, but the number of monitored channels has less impact on performance.

For this comparison, we did not consider the impact of the CPU clock, network capability, and memory capacity. A similar analysis has been done for pvAccess, and the results are shown in Table 9.

Table 9: pvAccess Performance Comparison (x86/amd64).

	CASE															
	2/1	4/3	2/4	1/3	2/1	4/3	2/4	1/3	2/1	4/3	2/4	1/3	2/1	4/3	2/4	1/3
CHANNEL	Delay = 0				Delay = 1				Delay = 5				Delay = 10			
1	0.86	1.02	0.87	1.03	0.78	1.02	0.77	1.00	1.05	1.06	0.97	0.98	1.05	1.04	0.97	0.97
10	0.93	1.20	0.77	0.99	0.87	0.98	0.83	0.93	1.04	1.05	0.96	0.97	1.03	1.03	0.97	0.96
100	1.25	1.39	0.35	0.39	0.72	0.94	0.77	1.00	1.03	1.05	0.94	0.96	1.02	1.03	0.92	0.93
1,000	0.93	1.21	0.59	0.76	0.59	1.02	0.58	1.01	0.78	0.98	0.71	0.89	1.02	1.03	0.92	0.93

Tables 2 and 3 show that array size has less impact to pvAccess performance on a 32-bit CPU or a 64-bit CPU.

IV. CONCLUSIONS

This benchmarking is carried under different client/server configurations. Three (3) test cases are evaluated. The current implementation of pvAccess provides flexibility for the user to configure a waiting time for the TCP package transport over the network. The reason for this is to avoid sending messages one by one. Instead of immediately sending a message, the server waits briefly for other messages to come. If no messages come, the server sends what it has after the waiting time has elapsed; otherwise, the server sends them as one batch. This improves performance in a "chaotic" environment. However, there are also (bad) side effects when you want minimum round-trip time... or if you have "controlled" environment (synthetic) tests.

The benchmarking was performed under a “controlled” environment, and the results under this benchmarking are as below:

1. When setting waiting = 0, it means sending any TCP package immediately, CA and pvAccess show similar performance.
2. However, CA shows better performance for processing get when the number of records is around **10.**
3. The x32/x64 CPU shows less impact to pvAccess performance, but large impact to CA. This conclusion is preliminary, and did not consider the impact of the hardware configuration, such as memory, CPU clock, network interface, and so on.

NOTE: The pvAccess used for this benchmarking is implemented in Java, and CA/EPICS is implemented in C/C++. A future benchmarking will be carried out once a C++ version of pvAccess is ready, which is under development actively, and will be available in Feb. 2011.

V. ACKNOWLEDGEMENT

The author wants to thank Matej Sekoranja from COSYLAB and Marty Kraimer for their fruitful discussions during this benchmarking.

VI. REFERENCE

1. http://sourceforge.net/projects/epics-pvdata/

Appendix I: Measurement Data for Array Performance

CA measurement

ARRAY	CASE 1	2	3	4
10	18,121	18,024	28,339	31,985
100	7,165	6,935	26,828	21,771
1,000	1,310	1,297	9,051	4,225
10,000	407	337	1,337	500
100,000	33	25	140	42
1,000,000	3	3	13	4

pvAccess measurement

	CASE															
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
ARRAY	Delay = 0				Delay = 1				Delay = 5				Delay = 10			
10	27,022	30,613	37,438	31,618	5,866	6,542	6,528	7,386	1,697	1,793	1,769	1,865	903	945	925	964
100	22,643	23,705	31,646	33,288	5,644	6,294	6,415	6,821	1,684	1,779	1,747	1,824	906	938	922	954
1,000	7,081	7,049	9,638	9,501	3,917	3,990	5,178	5,647	1,485	1,534	1,630	1,750	845	866	892	929
10,000	1,120	1,219	1,359	1,362	937	889	1,324	1,343	801	798	1,307	1,310	544	532	787	793
100,000	138	138	139	139	138	138	138	138	137	138	138	138	134	133	135	135
1,000,000	14	14	14	14	14	14	14	14	14	14	14	14	14	14	13	14

Appendix II: Measurement Data for Process Get Performance

CA measurement

CHANNEL	CASE 1	2	3	4
1	42,96	5,228	7,333	7,467
10	54,626	50,689	102,636	91,430
100	43,536	51,866	90,189	96,421
1,000	94,104	93,012	88,218	96,686

pvAccess measurement

	CASE															
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
CHANNEL	Delay = 0				Delay = 1				Delay = 5				Delay = 10			
1	4,113	4,695	5,355	6,087	676	751	743	817	176	186	181	190	92	96	94	97
10	25,619	28,177	34,374	30,704	5,898	6,540	6,485	7,427	1,723	1,799	1,776	1,864	913	943	929	964
100	60,782	55,418	67,142	100,567	24,592	27,740	28,365	38,500	12,422	13,356	14,556	15,278	7,600	7,930	7,957	8,644
1,000	82,296	82,281	87,058	90,892	65,255	73,815	54,847	58,192	48,678	49,852	46,856	47,760	7,600	7,930	7,957	8,644

Appendix III: Measurement Data for Event Processing Performance

CA measurement

	CASE			
CHANNEL	1	2	3	4
1	233	327	1294	1418
10	2,115	3,318	12,210	13,100
100	20,765	27,733	110,050	119,199
1,000	166,614	160,905	708,053	692,324

pvAccess measurement

	CASE															
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
CHANNEL	Delay = 0				Delay = 1				Delay = 5				Delay = 10			
1	214	184	208	212	212	165	212	216	170	179	174	185	89	93	92	96
10	3578	3342	3624	4349	3279	2857	3521	3437	1703	1765	1754	1834	894	925	930	956
100	80,342	100,121	205,480	286,359	39,314	28,501	39,319	36,845	14,868	15,240	15,498	16,214	8,078	8,273	8,719	8,954
1,000	5,059,908	4,704,893	6,640,922	8,028,389	157,433	92,435	156,443	159,846	144,164	112,166	16,1497	15,7966	8,078	8,273	8,719	8,954