The Performance and Throughput of OPC

A Rockwell Software Perspective

June 11, 1998

Thomas J. Burke

Rockwell Software Inc

ABSTRACT

The purpose of this white paper is to provide a framework for understanding the performance and throughput of OPC client and OPC server applications.

The focus of the study was to evaluate and measure the amount and how fast a server could serve data to client applications. Results documented in this paper are from Rockwell Software client and server applications. A sample server and sample client application is available as part of this study for other vendors to use to evaluate their respective performance and throughput.

The testing focused on the subscription based architecture of OPC, where the server acquires the data from the data source and sends the data to the client application when the value (or state of the value) changed from the last value sent to the client application.

General performance of the other OPC interfaces, including the client continuous polling / reading is a subject addressed by other white papers.

Since the beginning of OPC, there have been those people skeptical that a standard generic interface could be specified, designed, implemented and adopted by a number of vendors, and at the same time provide the level of performance and through-put of vendor specific proprietary data exchange mechanisms.

The architecture of OPC was designed to address the following:

1. Clients and servers residing on different physical computers(distributed)
2. Multiple clients communicating to a single server
3. A Single client communicating to multiple servers
4. Multiple clients simultaneously communicating to multiple servers

The OPC Foundation and the OPC Foundation member companies will focus on making sure that products are interoperable together, and be definition, vendors push the OPC Foundation to make sure that performance and throughput expectations are not compromised to achieve interoperability. Therefore, the Objects, Interfaces and functionality defined by OPC will continue to evolve and change as the technology continues to grow.

A side affect of this paper is to dispel the rumors and calm the critic's of OPC.

OBJECTIVE

The objective of OLE for Process Control (OPC) was to provide an industry-standard mechanism to communicate and exchange data between clients and servers by using leading edge OLE technology from Microsoft.

The goals of the OPC Task Force was for the standard to be widely accepted, adopted and implemented by all the client and server vendors as the primary mechanism for exchanging data.

OPC is really intended to solve the end user's problem of interoperability between multiple vendors' products. The end-user has the most to gain by vendor products working well together sharing data. The end-user expects that multiple vendor products work together, and in the past, client vendors had to develop their own suite of servers / drivers to meet the end-users expectation.

To achieve the objective and goals of OPC, we knew that we had to demonstrate that the performance and throughput of OPC exceeded the requirements and expectations of the end users of our products.

SUMMARY OF RESULTS

The results indicate that the amount of data that a server can serve to client applications whether the client is local or remote (using DCOM) exceeds the amount of data that realistically could be processed by a typical client application. In real world scenarios, using exception based notification, the amount of data that actually changes is typically a small percentage of the data that is being monitored.

From the start of OPC, there were those skeptics that doubted that a standard interface could be created and provide the level of performance of existing proprietary data exchange mechanisms.

The testing focused on making sure that the performance and throughput were deterministic (sustained and maintained over a period of time).

Tests were run where the client and the server were running on the same computer (called local), and tests were run with the client(s) and servers residing on different physical computers (called distributed or remote). We tested where the datatype of an item was a single word, as well as an array of words.

The following section includes some examples of the throughput performance tests that were run to characterize the performance and throughput of OPC. Results of additional performance tests are available in an excel spread sheet (RSOPCSUBPERF.xls), from Rockwell Software, or from the site you obtained this white paper.

To evaluate the performance of OPC in the distributed environment, we used six Pentium 266 computers, five computers running the client application, and one computer running the server application.

Each client application defined and added 10,000 items (AKA tags) to the server, and requested the server to update the client application at a rate of 250 milliseconds per item. We deliberately programmed the data source such that all the data was constantly changing (worst case scenario).

The result (see table below) was that the server was able to update two hundred thousand (200,000) items per second continuously.

	Items	Changes / second	Items / second
Client 1	10000	4	40,000
Client 2	10000	4	40,000
Client 3	10000	4	40,000
Client 4	10000	4	40,000
Client 5	10000	4	40,000
Server Total			200,000

For this test, each client application defined and added 100 items (AKA tags) to the server, with each item being 200 words of data. The datatype for the item was safe array of 4-byte signed integers (VT_I4 || VT_ARRAY).

The client requested the server to update the client application at a rate of 70 milliseconds per item.

The result (see table below) was that the server was able to update one million four hundred thousand (1,400,000) words per second continuously.

	Items	Words / Item	Changes / second	Items /second	Words / second
Client 1	100	200	14	1400	280,000
Client 2	100	200	14	1400	280,000
Client 3	100	200	14	1400	280,000
Client 4	100	200	14	1400	280,000
Client 5	100	200	14	1400	280,000
Server Total				7000	1,400,000

To evaluate the performance of OPC on a single (local) computer we used a Pentium 266 computer, running the both the client application, and the server application.

The client application defined and added 10,000 items (AKA tags) to the server. The datatype of an item in this case was a 4byte-signed integer (VT_I4)

The client requested the server to update the client application at a rate of 200 milliseconds per item.

The result (see table) was that the server was able to update fifty thousand (50,000) items per second continuously.

	Items	Changes / second	Items / second
Client 1	10000	5	50,000

For this test, the client application defined and added 100 items (AKA tags) to the server, with each item being 500 words of data. The datatype for the item was safe array of 4-byte signed integers (VT_I4 || VT_ARRAY).

The client requested the server to update the client application at a rate of 60 milliseconds per item.

The result (see table below) was that the server was able to update eight hundred thousand (800,000) words per second continuously.

	Items	Words / Item	Changes / second	Items /second	Words / second
Client	100	500	16	1600	800,000
Server Total					

OVERVIEW OF THE TESTING

The performance numbers in this white paper documents performance and throughput that may be achieved by applying the technology. Actual numbers may vary according to the implementation and what interfaces and functionality the respective OPC servers and OPC clients are using.

There were no tricks that would prevent any other OPC client application from working with the OPC server used in the tests, and no tricks preventing any other OPC Server working with the OPC client application used in the tests. The OPC clients and OPC servers used for the performance tests applied the OPC technology as specified in the OPC specification

In general performance and throughput are highly dependent on the hardware configuration, and for this type of performance study, the amount of data that can be acquired from the underlying data source.

I encourage other vendors to measure the performance on other hardware platforms, and to provide those numbers for other OPC Foundation members.

The focus was to evaluate the performance of the OPC functionality with respect to exchanging data between the OPC clients and the OPC servers developed by Rockwell software. Tests were run with single and multiple clients communicating to local and remote servers. The focus of the tests was to evaluate and measure the possible throughput that could be achieved using the standard OPC data exchange mechanisms.

Some of the same tests were also run with servers to simulate data change, and attempt to change more data than could be typically acquired from an external physical data source. In doing this we factored out any overhead associated with the acquisition and turnaround time associated with the physical device.

On the client side of the fence, the behavior of the client was to receive, process and display the data received.

For the DCOM testing, we used both 10BaseT network, and 100BaseT networks between the client and the server computer. As expected the throughput was significantly better using the 100BaseT network.

The typical usage of OPC will involve a server application updating multiple items simultaneously to multiple client applications. Our assumption was in a real application of the technology, the server would seldom only send a single item to a client, rather multiple items will be buffered and sent to the OPC client application in the construct of an OPCGroup. (This is the architecture and purpose of an OPCGroup and associated UpdateRate). Therefore we wanted to know how much data could be sent per second, rather than how long it takes for a server to send a data value to a client application.

The overhead associated with using the timestamp format was found to be negligible when the number of items sent in a single OnDataChange from the server to the client application was less than 500.

Future architecture will utilize capability of exception based notification at the data source, further reducing the amount of actual data that is needed to be processed by the OPC Server.

THE CLIENT APPLICATIONS

The client applications used for the testing were RSView32, and the Rockwell Software OPC Test Client application. The purpose of the Rockwell Software OPC Test Client application was to test OPC Servers. Both applications essentially exercised OPC the same way; they connected to the server, defined a group, and added items to the group. The group and the item were then made active. The client applications used the subscription architecture of OPC, where the server sends the data to the client application in the exception mode (where the value or state of the value has changed).

We validated that the throughput the server could send to either client application was close to equivalent, and focused on making sure that the performance and throughput was deterministic, repeatable and clearly measurable. The assumption we made is that general client architecture would have a dedicated thread for receiving the data, and other threads dedicated to processing or displaying the received data. The results (even for a lot of items) clearly demonstrate that the client is able to update a display faster than the eye can detect.

RSView32 is an off-the-shelf, easy-to-use, Windows®-based system with all the features you need for effective monitoring and supervisory control. It's the first MMI to embed Microsoft Visual Basic for Applications to allow customization of applications and interoperability with other applications. RSView32 is also one of the first MMI software in the industry to be both a client and server of OPC data. This provides users with added flexibility for peer-to-peer networking as well as the ability to create a reliable system that includes control products from multiple vendors.

THE SERVER APPLICATIONS

The server applications used for the testing were RSLinx, and the Rockwell Software OPC Emulator application.

Data from the client's perspective was changing faster than the client wanted to receive the data values essentially every scan the value had changed from the previous value.

On the server side of the equation, we deliberately used the RSLinx OPC Server for most of the throughput testing. We wanted to factor in the capacity for the server acquiring the data from a physical device, processing the data, evaluating for change of state, and sending the data to the client application(s) into the equation.

With the simulated server, we eliminated the overhead associated with server acquiring the data from a physical device, processing the data, and evaluating for change of state, and as expected actual throughput increases.

RSLinx for Allen-Bradley Programmable Controllers is a complete factory communications solution for the Microsoft® Windows 95/NT® operating system. It provides a means for data exchange between a PLC® processor and a wide variety of client applications including Rockwell Software's RSView32, RSLogix, RSTools and RSSql™. RSLinx communication capabilities have now been extended to support OPC connectivity to maximize the interoperability between clients and servers.

FOR MORE INFORMATION

Rockwell Software will provide a copy of the sample server and sample client executables used for the throughput testing for other vendors to benchmark the performance and throughput on other hardware platforms. We encourage those vendors to provide that information to Rockwell Software, for inclusion in future releases of the performance and throughput spreadsheet (RSOPCSUBPERF.xls).

Thomas J. Burke

(440) 646-7727 (voice)

mailto:Thomas.Burke@software.rockwell.com

http://www.software.rockwell.com
http://www.opcfoundation.org