DCOM, OPC and Performance Issues

Al Chisholm, Intellution Inc

2/3/98

Abstract

There have been persistent and totally unfounded rumors about performance problems with Microsoft DCOM and OPC. We suspect that most of these stories are at best 'wishful thinking' on the part of vendors who have fallen seriously behind the curve in terms of technology or who are locked into obsolete technologies. This brief paper presents some hard test results that will show the true performance of OPC and Microsoft DCOM.

Summary of Results

As you will see from the numbers below, OPC and DCOM performance are more than adequate for the vast majority of both dedicated and distributed applications when running on commonly available computers such as a P233. Server performance is particularly impressive in that a P233 server was able to supply nearly 20,000 values per second to 4 clients with only 10% CPU load. In real world implementations where exception based notification is used it is likely that CPU use by the OPC/DCOM component will be even less.

We were also impressed by the fact that OPC and DCOM were able to provide 300+ transactions per second for the single item requests. This implies that small, simple, non-optimized applications as might be created for a specific task can also be used effectively with OPC.

It appears that DCOM and OPC impose a somewhat higher overhead on older machines such as a P120 but even here the results would be acceptable in most applications since the 'worst case' scenario still allows such a machine to obtain 4000+ values per second with only 20% CPU use.

You will see that the InProcess model allows for the highest performance. This would typically be used in a smaller system or between a SCADA, Softlogic or DCS engine and an I/O driver on one node of a larger multi tier distributed system. This model allows 10000+ values to be transferred per second using less than 1% of the CPU. This is clearly more than enough throughput for any conceivable application.

Overview of the Test

As you will see from the discussion of the test client and server programs, this test is specifically designed to show the total overhead that is specific to COM, DCOM and OPC. It omits any vendor specific overhead such as COMM port or Device Network management on the server side or screen or database management on the client side. The tests were run on In-Process, Local and Remote servers. The Remote tests were run with 1 and with 4 clients. The test results include the machine speeds and CPU use. All machines were running NT 4.0 Workstation. The network was a very busy 10BaseT network that was being shared with several hundred additional workstations.

In several cases, tests where run with both multi item and single item reads. Multi item reads give an accurate assessment of how OPC will perform in practice. Single item reads allow us to investigate the transaction turnaround time and thus help gauge the performance of the underlying COM and DCOM protocols. In practice single item reads would seldom be used by a real application.

Also keep in mine that for each Data Item, OPC transfers a Timestamp and Quality Mask along with the value. The numbers below for 'per Item rates' in fact count all 3 of these components as a single item. Other technologies such as DDE would need to transfer 3 times as many data items to get the same amount of information back to the client. Keep this in mind when reviewing throughput numbers for other technologies.

The Client Program

The client is a program called OPCSPEED.EXE. It was written specifically to evaluate OPC performance. For purposes of this test we are doing synchronous reads which is essentially a 'worst case' scenario. The client program creates OPC Groups, adds items and reads floating point data values. It also performs all needed COM memory management. Thus this client should accurately reflect the OPC and COM specific overhead in these tests.

Note that in practice many applications use their OPC Servers in 'exception' mode where only the data that has actually changed is transferred. Thus the numbers obtained from this test indicate the maximum rate at which values can be transferred between client and server. The number of values that can be monitored effectively using OPC in exception mode is clearly far higher.

The Server Program

The server in this case is the OPC Sample Server provided by the Foundation and written by Al Chisholm of Intellution. The code in this server is a simple but very complete implementation of the COM and OPC interfaces including simulated data generation. A complete server would need to add only the device or vendor specific communications logic to replace the simulated data with real data. Thus this server should accurately reflect the combined OPC and COM specific overhead in these tests.

Note that the server uses an extremely simple single threaded implementation and still achieves excellent performance with multiple clients. It is likely that an optimized, multithreaded implementation could attain even better performance.

The Numbers

This section presents the 'raw numbers'. You will see that in the InProc and Local tests we purposely set up the test to use all available CPU so as to ascertain the maximum possible throughput.

In Process Server -1 Client

	Speed	Items	Reads	Time (sec)	CPU Use	Items /sec	Transactions /sec
Client	P233	100	100000	9	100*	1111000	11000
Server	n/a						

Client	P233	1	5000000	22	100*	227000	227000
Server	n/a						

*Note for this test the client is placed in a tight loop in order to obtain maximum throughput. What this shows is that for a more typical throughput of 5,000 items per second for an InProcess Server Model, CPU use would be less than 0.5%.

Local Server -1 Client

	Speed	Items	Reads	Time (sec)	CPU Use	Items /sec	Transactions /sec
Client	P233	100	10000	16	40*	62500	625
Server					60*		
Client	P233	1	50000	15	50*	3333	3333
Server					50*		

*Note for this test the client is placed in a tight loop in order to obtain maximum throughput. What this shows is that for a more typical throughput of 5,000 items per second for a Local Server Model, total client and server CPU use would be approx. 8%.

Remote Server -1 Client

	Speed	Items	Reads	Time (sec)	CPU Use	Items /sec	Transactions /sec
Client	P233	100	1000	15	10*	6,666	66
Server	P233				12		
Client	P233	1	3000	9	10*	333	333
Server	P233				10		

*Note for this test the client is placed in a tight loop in order to obtain maximum throughput. The node to node transaction time acts as a gate on performance in the Remote Server Model. What this shows is that for a typical throughput of 5,000 items per second for a Remote Server Model, client CPU use would be approx 7.5% and Server use would be approx. 9%.

Remote Server - 4 Clients

	Speed	Items	Reads	Time (sec)	CPU Use	Items /sec	Transactions /sec
Client 1	P233	100	1000	23	7	4348	43
Client 2	P266	100	1000	22	4	4545	45
Client 3	P120	100	1000	17	25	5882	59
Client 4	P120	100	1000	25	20	4000	40
Server	P233				10	18775**	187**

**Note that this includes total server throughput for the combined 4 client load. This test shows that even a simple single threaded server is capable of providing nearly 20,000 data points per second to 4 clients with approx. 10% CPU use.