Performance Evaluation of OPC-based I/O of a Dynamic Process Simulator

Jyrki Peltoniemi, Tommi Karhela, Matti Paljakka
VTT Automation, Technical Research Centre of Finland
P.O.Box 1301, 02044 VTT, Finland
email: Jyrki.Peltoniemi@vtt.fi

Keywords: OPC, Performance evaluation, Process control, Automation design

ABSTRACT

As the use of dynamic process simulation in industry has increased, a need has emerged for an easy way to configure reliable and powerful data exchange between simulation applications and distributed control systems (DCS's).

A system where the control applications and man-machine interfaces of the real DCS are connected to a simulation model of the process, can be used for building, tuning and testing the control system implementation, and for training the operators, independently from the plant hardware.

OPC (OLE for Process Control) is a set of component specifications widely used in the automation domain for software application interoperability. OPC Data Access interfaces have been used e.g. for connecting DCS applications of multiple vendors to simulation models.

This paper discusses architecture and design issues that affect the performance of the OPC data exchange in applications involving simulation. The performance test results of an OPC server designed according to the guidelines given show that OPC-based communication can offer sufficient data transfer capacity for most dynamic process simulation purposes.

INTRODUCTION

Dynamic process simulation has long been used in automation design and testing. For these purposes, it is often desirable to connect the simulation model to control application in a real DCS. If both systems support OPC data access specifications, the connection can be made using this technique. The concept has already been used for automation testing and for operator training. [1,2] The OPC-based connection has proved to be an easy-to-configure tool, but it lacks the ability to manage larger scale use. For instance, the transfer capacity of OPC servers has been too small for transferring a set of items comprising several thousands to tens of thousands values a second.

The aim of this paper is to discuss the features that affect the performance of OPC data access based communication. The practical work has centered on the dynamic process simulator Apros (Advanced PROcess Simulator). Product has been developed by Technical Research Centre of Finland (VTT) and Fortum, and is available for modeling of combustion power plants, nuclear power plants and pulp and paper mills.

The motivation for developing OPC was to find an easy way to communicate between numerous data sources in the field of automation. The costs of integrating equipment from different vendors can be significant. A considerable amount of engineering time may also be required in writing drivers for a typical supervisory-control software project. [3]

OPC provides a plug-and-play software technology for automation- and process control industries. It has rapidly attained worldwide de-facto standard status. One reason for this is that the OPC specification is built on the existing, widely used binary standard COM. [4]

The OPC specification includes Data Access, Alarms & Events, Batch, Historical Data Access and OPC XML specifications. The most widely used is the Data Access (DA) specification. This paper mainly covers issues related to data access specification.

As mentioned, OPC specification is build on COM. A DA OPC client can use either a custom interface or an optional automation interface. Figure 1 illustrates the situation.

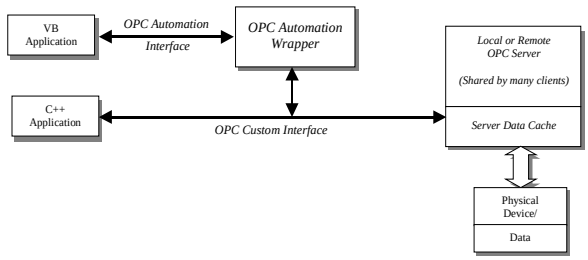

Figure 1. Typical OPC architecture [3]

The data access specification consists of two main objects: the OPC server and the OPC group object (Figures 2-3). Every OPC client creates its own OPC server object,

which can contain several group objects. The group holds items that the client wants to observe or manipulate. Items are created when the client wants to observe or manipulate a specific variable in the device. Every item has an item ID, which is unique within the OPC server. Every variable can be a member of several groups. A unique item id is used when creating an item. However, read and write operations are done with the help of an OPC server generated group-specific server handle and an OPC client generated client handle. Because the OPC server decides which item handles it provides to the client, these handles can be used to provide constant time access to each item in the database of the OPC server.

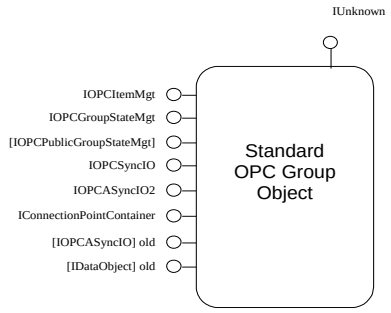

Figure 2. Standard OPC group object [3]

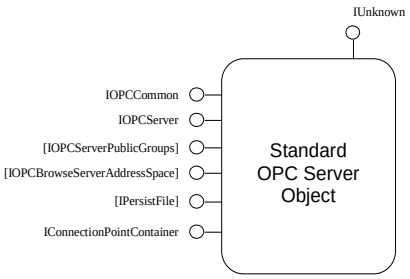

Figure 3. Standard OPC server object [3]

CONNECTING PROCESS SIMULATOR AND DISTRIBUTED CONTROL SYSTEM

One of the most interesting uses of the OPC data access specification is making a connection between a dynamic process simulator and a distributed control system. This kind of combination can be used for operator training and automation design and testing purposes [1,2].

If two or more applications exposing OPC interfaces have to communicate with each other, a cross connector application can be used to connect the OPC servers. Figure 4 illustrates the situation.

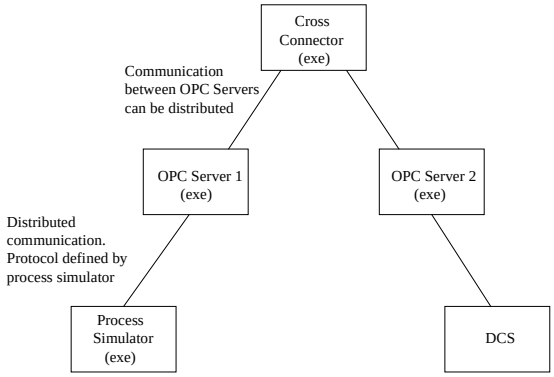

Figure 4. Current architecture for connecting a DCS and a dynamic process simulator

The current OPC server implementation for Apros is a stand-alone application. This kind of approach, where the OPC server uses a native socket-based communication to contact a database of a device, is widely employed. The server implementation can adopt the benefits of an existing legacy communication protocol. However, this kind of architecture means that extra components have to be introduced even when only two devices are connected. One benefit can be that, when functionality of the device is distributed, the same OPC server can contact all the required devices, thus hiding the distributed nature of the system.

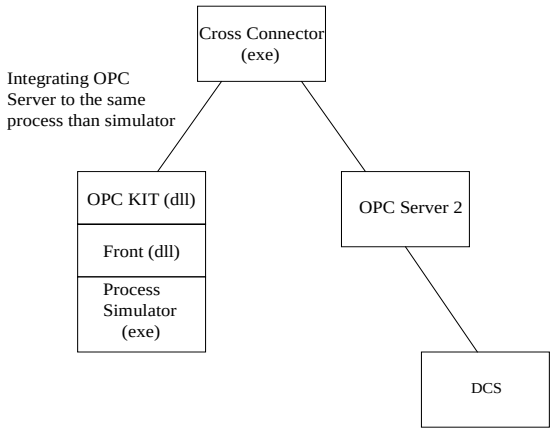

Figure 5. Integrating OPC server into the simulator

A data access OPC server implementation for more compact data sources, like a dynamic process simulator in our case, can be integrated into the same process as the actual data source. This can lead considerably more effective data exchange. Figure 5 describes the new architecture where the OPC server of the simulator consists of two dynamically linked libraries. Next section will consider more detailed information about the design principles of the new architectural approach.

One possibility to simplify a connection between two OPC servers is to implement cross-connector functionality directly at either end of the communication. However OPC specification does not define a standard OPC client. This means that the embedded cross connector has to be configured in a vendor-specific way (Figure 6).

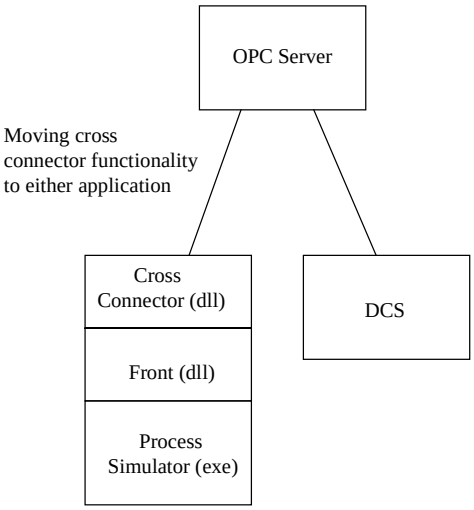

Figure 6. Integrating cross connector application into the process simulator

One architectural approach to the OPC server implementation is to use an existing communication protocol of a legacy system and implement the OPC server as an in-process server. However this may not be the most elegant way and cannot be as efficient as the integrated solution (Figure 7).

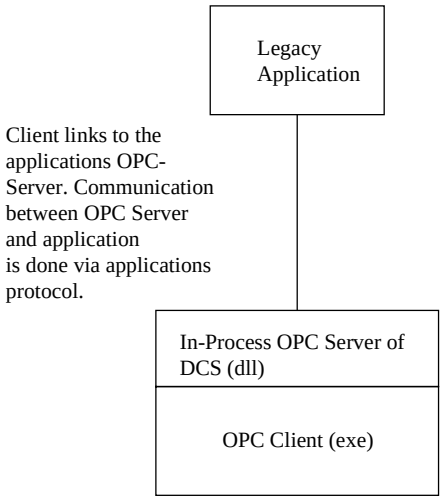

Figure 7. In-process OPC server communicating with an existing protocol

MAIN DESIGN PRINCIPLES

The most important criterion for new OPC server, based on the architecture illustrated in Figure 5, was efficient data exchange. The simulation software is a rather old program and can be considered as a legacy system. This was one of the main reasons why the OPC server design consists of two libraries (Figure 8). OPC Kit I/O library contains all OPC specific code. Between I/O libraries and simulator is Front library. When the simulator is used on a platform other than Windows, other I/O libraries can be built on top of the Front. Also the integrated cross connector I/O library can be built on top of the Front. The interface between the Front and I/O libraries was designed to be simple and not to include OPC-like generality and OPC specialties.

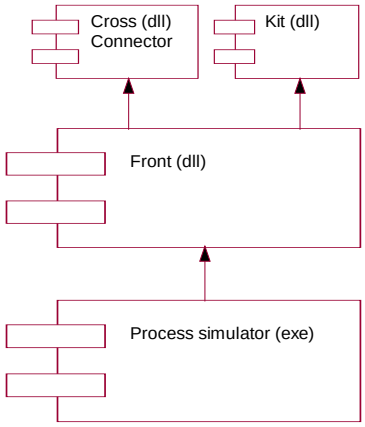

Figure 8. New OPC server design

If new simulation engines are built, they can implement the general purpose C interface between the Front and I/O libraries, and reap the benefits of the OPC Kit without any extra work.

The Front component can be used to make variable mappings to higher level variables and an intelligent Front component can map closely related variables for example to vectors. OPC standard and the Kit library support one-dimensional vectors. However scalar items cannot be mapped or collected dynamically to vectors through OPC interface.

The most efficient OPC server implementation avoids any extra data caching. Using an extra component between I/O libraries and device does not necessarily mean that the data have to be copied to the Front. If the data types of the target simulator are identical to the data type definitions in the Front-Kit interface, references to the data can be forwarded directly to the Kit. This kind of implementation uses the Front as a stateless component that only controls the data exchange. If the data types cannot be mapped one-to-one, the data have to be copied and converted to another format. If a more robust design is essential, access to the

database of the simulation server can be restricted to the Front. However, this is done at the cost of efficiency.

MEASURING THE PERFORMANCE

Performance measurements were made with a laptop PC (500MHz Mobile Pentium III and 320-MB RAM). The operating system was Windows NT 4 Workstation with service pack 6. In all tests OPC server was connected from different process in the same machine. DCOM performance was not measured. Data exchange is a highly CPU-intensive task. This suggests that, in the near future, the performance of data-transfer applications will be significantly improved.

In order to illustrate how the enhanced performance of processors will affect the throughput of the OPC server, the performance values achieved with a PIII 500MHz are compared with the values achieved with an AMD Athlon 1.2GHz and 256-MB RAM. However, the operating system was Windows 2000.

The performance of an OPC server often depends of the underlaying device-connection more than raw OPC. Chisholm has discussed OPC performance without any real system behind [5].

In typical process simulation case it is more important to achieve the biggest throughput possible rather than very fast responses. Event based communication, which is implemented with connection points according to the data access 2 specification, is suitable for this purpose. The client subscribes a set of items and determines the frequency needed. In following tests all double items were continuously changing and the requested frequency was 200ms. Table 1 summarizes the results. The complexity of the algorithm used in event based data change is linear O(N). The OPC client used in test did nothing with the received values. The first column makes the difference between old and new implementations. The fourth column shows the CPU load of the OPC server. CPU Simulator shows how much resource has been used to the background simulation.

Next results consider writing to the database of the simulation engine. The most efficient and the simplest way for manipulating the database is through the synchronous write. Performance metrics (Table 2) was achieved by simply measuring the time consumed in IOPCSyncIO->Write function. Tests were made for the OPC Server based on the new architectural approach. The complexity of the write operation is O(N).

In the previous sections, the concept of connecting two or more devices with a cross-connector application was introduced. We used a test case in which two process simulators were connected with existing cross-connector software. Figure 9 shows the architecture.

Table 2. Performance of synchronous write

	Processor	Items	Time (ms)
new	PIII 500MHz	1	1.3
new	PIII 500MHz	1000	2.8
new	PIII 500MHz	10000	29
new	PIII 500MHz	50000	160
new	AMD 1.2GHz	1	1.5
new	AMD 1.2GHz	1000	2.3
new	AMD 1.2GHz	10000	27
new	AMD 1.2GHz	50000	120

In order to keep the impact of the simulation time as low as possible, sine waves were used to generate continuously changing items. The first simulator simulated sine signals,

Table 1. Performance of event based data exchange 2. * denotes that the processor time consumed for simulation is included in CPU server column. This is a case when OPC server is integrated into simulation software. CPU times consumed are divided into privileged, user and total time. proc. denotes processor.

	Processor	Items	CPU priv.	% user	servr tot.	CPU priv.	% user	client tot.	CPU priv.	% user	proc. tot.	CPU % simulator
old	PIII 500MHz	5000	1	41	42	1	4	5	3	49	52	5
old	PIII 500MHz	8000	1	70	71	1	5	6	6	80	86	5
new	PIII 500MHz	5000	1	9	10	1	4	6	2	13	15	*
new	PIII 500MHz	10000	1	19	20	1	5	6	3	24	27	*
new	PIII 500MHz	30000	2	46	48	10	13	23	26	59	85	*
new	AMD 1.2GHz	5000	0.1	5	5	0	0.1	0.1	1	5	6	*
new	AMD 1.2GHz	10000	1	9	10	1	4	5	7	13	20	*
new	AMD 1.2GHz	30000	1	33	34	5	10	15	13	43	56	*
new	AMD 1.2GHz	50000	2	56	58	10	14	24	26	70	96	*

Table 3. Performance of two simulators connected with a cross-connector application. Requested frequency was 200 ms.

	Processor	Items	CPU % C-Con	CPU % servers	CPU % proc. priv.	user	total
old	PIII 500MHz	2500	6	45	7	50	57
new	PIII 500MHz	2500	6	12	8	17	25
new	PIII 500MHz	5000	19	17	10	30	40
new	PIII 500MHz	9000	38	27	21	59	80
new	AMD 1.2GHz	9000	22	24	9	41	50
new	AMD 1.2GHz	11000	39	37	17	64	81

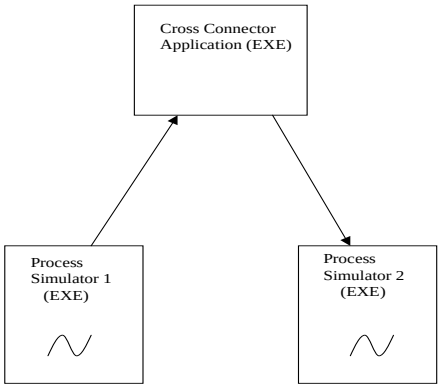

Figure 9. Connecting two process simulator with a cross connector application

which were sent with event-based data exchange to the cross-connector application. The frequency of the data exchange was 200 ms. The CPU resources used in the first simulator, which was used to simulate sine waves, were about 5%. The cross connector was writing all the values to the other simulator. The results show the impact of the non-optimized cross-connector, which clearly becomes the bottleneck of the system. Coding the functionality of the cross connector on top of the Front library, as discussed earlier, can increase throughput considerably.

The result shows how important it is to optimize all parts of the system when maximum performance is required. This can be done most efficiently by reducing the number of components in the overall system.

CONCLUSIONS

OPC data-access based communication has earlier been used for connecting a DCS and a dynamic process simulator in small- to medium-sized projects. A better performance can help to take the step towards larger scale use. Using standard means of communication can reduce the costs of system integration. Although a better performance can still be achieved using a block-based data transfer, it seldom offers such a flexible and easy-to-use behavior as OPC.

Reducing complexities of the OPC server design and integrating server functionality into the same process with simulation software, would provide the speedup needed for large-scale use of dynamic simulation. When high performance is required in the combination of a DCS and a dynamic process simulator, both OPC servers must be optimized.

The amount of simulation aided automation deliveries in the process industry will grow in the future. Tendency towards open architectures is continuing and communication protocols are developing. Tools for the simulation-aided working methods are evolving into a more scalable and effective form for the benefit of the end-user.

REFERENCES

[1] Lappalainen, Jari; Tuuri, Sami; Karhela, Tommi; Hankimäki, Janne; Tervola, Pekka; Peltonen, Soile; Leinonen, Toivo; Karppanen, Erkki; Rinne, Jarmo; Juslin, Kaj. "*Direct Connection of Simulator and DCS Enhances Testing and Operator Training*". Proceedings of TAPPI 1999 Engineering / Process & Product Quality Conference, Hilton Anaheim, Anaheim CA, September 12-16 1999, p. 495-502

[2] Rinta-Valkama, Jarno; Välisuo, Martti; Karhela, Tommi; Laakso, Pasi; Paljakka, Matti. "*Simulation Aided Process Automation Testing*". IFAC's Conference on Computer Aided Control System Design (CACSD), University of Salford, UK, September 11 - 13 2000.

[3] OPC Foundation, "*OPC Data Access Custom Interface Specification*", 2.04 ed., September, 2000.

[4] Microsoft Corporation, "*The Component Object Model Specification*", 0.9 ed., October 1995.

[5] Chisholm, Al, "*DCOM, OPC and Performance Issues*" Tech. Report, OPC Foundation, March, 1998.