Going Beyond Performance Limitations of OPC DA Implementation

Yauheni Veryha
ABB Corporate Research
Wallstadter Street 59
68526 Ladenburg, Germany
yauheni.veryha@web.de

Abstract

One of the most important factors in building high-performance, scalable industrial control applications is the ability to transfer process data from the server to the client machine in the most efficient and fast way. OPC DA is widely accepted and used as the standard in automation and network control systems. We present a system and method for caching process data in large industrial control systems to overcome performance limitations of the OPC DA implementation. Our approach for large network control systems includes a powerful and easy-to-use client data cache that allows overcoming typical performance limitations of the OPC DA implementation for large data transfers.

1. Introduction

Caching is a classic, successful technique to improve the performance of software programs by exploiting temporal references and providing high-bandwidth, low-latency access to cached data [1]. In the client - server architecture of software programs, the data cache is typically implemented on the server side, because it is simpler to implement, easier to synchronize and maintain [4]. All programs running on the client machines may equally use server-side cache whenever they require a fast access to the process data.

The main drawback of the server-side data caching shows up if client software programs running on the client machine require large amount of process data. The data transfer from the server-side cache to the client machine may become a performance bottleneck, as it is the case with client-server applications using OPC DA for large data transfers (e.g. more than 50,000 data points) [8]. The use of the client data cache becomes an option in such situations.

Client-side data caching is widely used in DBMS (Database Management Systems) and Web applications. One of the reasons for creating client data cache in DBMS is to make persistent state information from the server available locally to the client for data-intensive processing [10]. This way, the client application does not need to make repeated invocations to the server application to retrieve data from the server cache [4].

In Web applications, by moving components of the database system to the web client, user access times can be improved by reducing both server load and communication costs [11]. General web caching strategies typically do not cache dynamically generated documents, nor deal with the case of client-side applications retrieving query results from a database on the server side [5]. In the work [3], the client-side caching framework that may optionally contain a personal assistant is presented. The personal assistant monitors user's access behaviour to dynamically pre-fetch data in accordance with the predicted access patterns. Another aspect mentioned in [7] is the cache coherency that is ensured in cooperation with the server-side database agent where communication is established through the use of persistent socket connections between the client-side agents and the database agent. Changes to the database can therefore be propagated to all client-side agents ensuring that only current information is stored in the client agent's caches [7].

2. Instructions

Network control systems generally perform different functions such as HMI (Human Machine Interface), data logging, advanced control, I/O operations, etc. Personal computer systems running various operating systems, like Windows XP, Linux versions and the like are increasingly being used in the industrial control applications due to the open PC (Personal Computer) architecture, acceptable computational performance at a relatively low cost, a wide variety of off-the-shelf software and hardware products which provide a range of data acquisition, analysis, presentation, and management tools as well as a relatively easy connection of various computer systems within a network [2].

In the network control systems, there are different approaches for implementing communication between various software components located on multiple computer nodes in client-server architecture. One of the most popular standards today is the OPC DA (OLE for Process Control Data Access) [8]. OPC DA interfaces

0-7803-9402-X/05/$20.00 © 2005 IEEE

together with the DCOM (Distributed Component Object Model) are used to establish a communication between network control system components running on various computer nodes. The OPC DA server exposes a set of the OPC DA interfaces that have functions to browse, read and write variable values. These functions are marshalled across the network using, for example, Microsoft DCOM, so that OPC DA clients are able to call them from the client machine. OPC DA has a disadvantage that there is no mechanism for data compression and, thus, relatively complex binary data are normally transferred over the network. This is one of the reasons, which may create performance problems for large data transfers [8].

In large network control systems, client machines can subscribe to thousands of data points from server machine that interface various I/O devices. For a graphical presentation of this large amount of process data on the client machine, for example, to start a "world view" picture of 50,000 – 100,000 process objects in the process viewer, a large data transfer from the server to the client machine may consume a large amount of the network bandwidth, operating system resources, etc and, thus, significantly degrade system operation. This may lead to a situation when operators on the client machines will have to wait 3 – 5 minutes until all process data are finally available during the start of the "world view" picture in the process viewer. Therefore, additional improvements for large network control systems that use OPC DA are required to reach an acceptable system performance.

An example of the network control system, where the client data cache to improve the overall system performance is used, is shown in Figure 1. The exemplary network control system (see Figure 1) comprises various hardware I/O devices, and client and server computer machines that are interconnected through a network. Server computer system interfaces to various hardware I/O devices, such as PLCs (Programmable Logic Controllers), RTUs (Remote Terminal Units), etc. Each of the client computer systems preferably executes industrial control software that provides a HMI to the process. When a hardware I/O device generates a new data, a server process is notified and receives the data from the hardware I/O device. The server process with data access server has a list of clients who are subscribed to the server data generated by the respective hardware I/O device. In response to receiving the data element from the hardware I/O device, the server sends changed data over the network to a respective client, or to each of a number of clients, which are interested in the data. The server uses a socket-based communication to transfer data over the network to client machines.

Figure 2 illustrates a part of the network control system with the client data cache. As it is shown in Figure 2, one or more client machines may communicate simultaneously with the server machine.

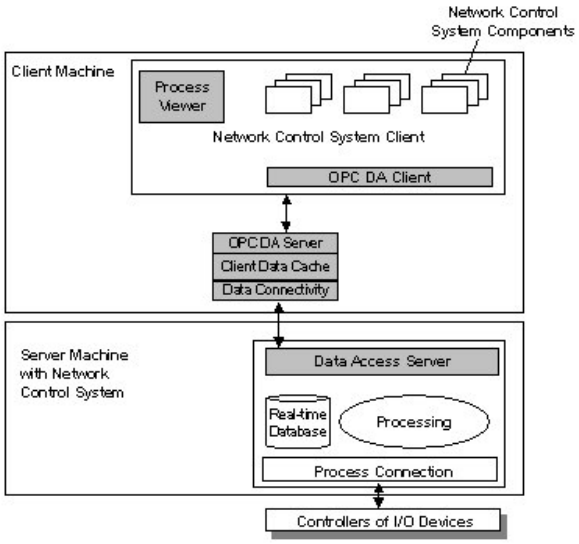


Figure 1. Client-server architecture of the network control system with the client data cache and the OPC DA implementation on the client machine.

The data access server of the network control system running on the server machine gets process data from the process connection module. The process data are presented as process variables in the data access server. The client data cache using proprietary socket-based connectivity implemented in C++ is able to subscribe to server process data. Process data from the data access server are stored in the client data cache as variables. The OPC DA interface of the client data cache makes it possible to present cache variables as OPC items for other OPC DA clients of various network control system components.

The supervisory manager of the client data cache offers features that allow customizing how items are cached and how long they are cached. It is possible to instruct the supervisory manager of the client data cache to give certain items priority over other items when the supervisory manager performs a removal of seldom used or unimportant items to free some memory.

3. Client data cache

The client data cache uses keys paired with values to present the data. All items contained in the client data cache are further available as OPC items using OPC DA interface provided for the client data cache (see Figure 2). Therefore, all components of HMI, like process viewer, faceplates, workplaces, etc with the OPC DA clients can subscribe to OPC client data cache values. Such an approach is especially important for HMI components that must be able to subscribe to a large

amount of process data, for example, to show "world view" picture of the given process to users. The data availability will be largely improved due to the lack of data transfer over the network between the client data cache and client HMI components.

Figure 3 illustrates the architecture of the client data cache with the OPC DA layer. The client data cache includes a list of variables, which are mapped to the variables in the data access server and OPC items in the OPC DA server layer. Appropriate get() and set() functions executed under the control of the supervisory manager allow keeping contents of client data cache, data access server and OPC DA server synchronized and updated. The supervisory manager includes features that allow customizing how items are cached and how long they are cached. It is possible to instruct supervisory manager of the client data cache to give certain items priority over other items in the client data cache when the supervisory manager performs removal of items. This can be done during adding of an item into the client data cache. One can also establish an expiration policy for an item when one adds it to the client data cache. Once an item expires, it is removed from the client data cache by the supervisory manager.

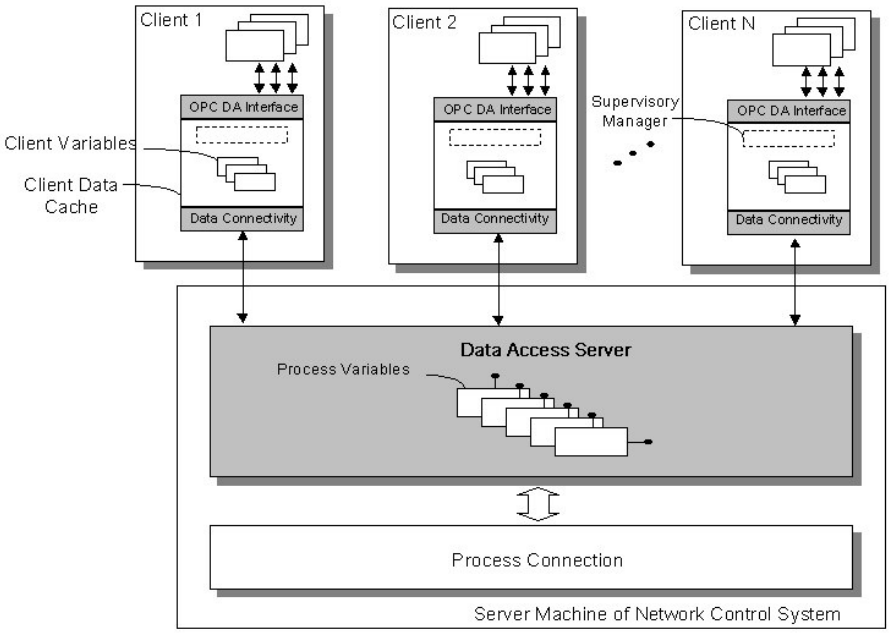


Figure 2. Client data cache in the network control system with the client-server architecture.

4. Performance

To compare the performance of the client data cache based approach with the conventional OPC DA implementation, we performed a number of tests by varying the number of variables in the data access server and OPC DA server respectively. The main test results are summarized in Table 1. The setup for the conducted performance tests with the client data cache consisted of the client and server PCs with Windows 2000 Service Pack 4, CPU ~2 GHz, RAM ~1 GB and hard disks with at least 20 GB of free space. The client and server PCs were connected via a 10Mbit/s Ethernet network. The detailed description of the setup for performance tests with the normal OPC DA implementation can be found in [8]. As one can see from Table 1, the performance of our solution with the client data cache and socket-based communication between the server and the client is more than twenty times higher than that with the normal OPC DA implementation.

Table 1. Comparison of performance characteristics for transmission of all variables from the server to the client.

Number of variables	Normal OPC DA implementation (Time in ms)	Client data cache (Time in ms)
100	240	9
500	1080	21
1000	2200	38
5000	4360	169
10000	7520	298
20000	12800	591

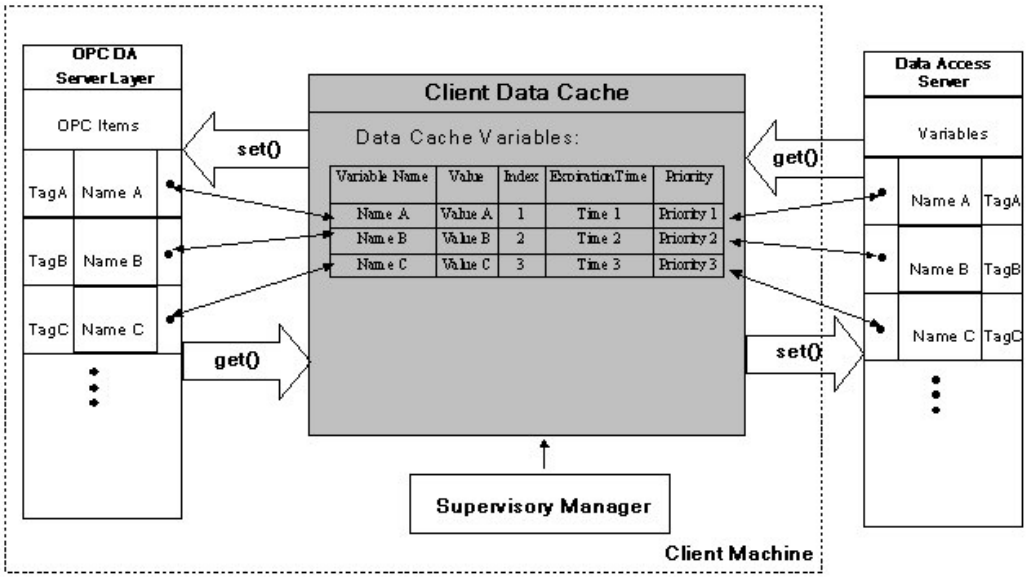


Figure 3. The architecture of the client data cache with the OPC DA layer.

5. Conclusion

We have presented an approach for performance improvement in industrial network control systems that includes a powerful and easy-to-use client data cache overcoming typical performance limitations of the OPC DA implementation for large data transfers. The main advantages of using the client data cache in the network control systems are:

- The improved data availability (2 to 20 times faster data transfer from the server to the client comparing to the normal OPC DA with DCOM) because of simpler ("flat") binary data structures and a simple socket-based communication,
- Easy integration with 3rd party applications that require process data on the client machine through OPC DA (Client data cache includes the OPC DA server on the client machine),
- No need for DCOM (DCOM has documented problems [8]), because data transfer is done using simple socket-based communication,
- The improved support of redundancy for fault recovery can be obtained on the client machine due to the storage of the latest process data in the client data cache.

The main drawbacks of the presented approach with the client data cache are:

- An additional overhead with client-server data synchronization,
- Additional maintenance costs may arise.

Future works will include an advanced testing of the client data cache in the large network control system and further improvements of the data synchronization algorithm.

References

[1] L. Bass, P. Clements and R. Kazman, *Software Architecture in Practice*, Addison Wesley Longman, 1998.

[2] S. Boyer, *SCADA: Supervisory Control and Data Acquisition*, ISA - The Instrumentation, Systems, and Automation Society, 1999.

[3] P. Cao and C. Liu, "Maintaining Strong Cache Consistency in the World Wide Web", *IEEE Transactions on Computers*, Vol.47, pp. 445-457, 1998.

[4] T. Carpenter, et al., "Client-Server Caching with Expiration Timestamps", *Distributed and Parallel Databases*, Vol.10, pp. 5-22, 2001.

[5] J. Hunter, W. Crawford and P. Ferguson, *Java Servlet Programming*, O'Reilly & Associates, 1998.

[6] M. Franklin, "Client Data Caching. A Foundation for High Performance Object Database Systems", *The Kluwer International Series in Engineering and Computer Science*, No.354, 1996.

[7] P. Gopalan, et al., "Caching with Expiration Times", *In Proc. of 13th ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, USA, pp. 540-547, 2002.

[8] F. Iwanitz and J. Lange, *OLE for Process Control*, Huethig, 2001.

[9] D. Patrick and S. Fardo, *Industrial Process Control Systems*, Delmar Publishing, 1997.

[10] G. Reese, *Database Programming with JDBC and Java*, O'Reilly & Associates, 1997.

[11] P. Sistla, O. Wolfson and Y. Huang, "Minimization of Communication Cost Through Caching in Mobile Environments", *IEEE Transactions on Parallel and Distributed Systems*, Vol.9, pp. 378-390, 1998.