USING SERVERS TO ENHANCE CONTROL SYSTEM CAPABILITY[*]

M. Bickley, B. A. Bowling[#], <u>D. A. Bryan</u>[+], J. van Zeijts[&], K. S. White, S. Witherspoon,
Thomas Jefferson National Accelerator Facility, Newport News, VA

Abstract

Many traditional control systems include a distributed collection of front end machines to control hardware. Back end tools are used to view, modify, and record the signals generated by these front end machines. Software servers, which are a middleware layer between the front and back ends, can improve a control system in several ways. Servers can enable on-line processing of raw data, and consolidation of functionality. In many cases data retrieved from the front end must be processed in order to convert the raw data into useful information. These calculations are often redundantly performed by different programs, frequently offline. Servers can monitor the raw data and rapidly perform calculations, producing new signals which can be treated like any other control system signal, and can be used by any back end application. Algorithms can be incorporated to actively modify signal values in the control system based upon changes of other signals, essentially producing feedback in a control system. Servers thus increase the flexibility of a control system. Lastly, servers running on inexpensive UNIX workstations can relay or cache frequently needed information, reducing the load on front end hardware by functioning as concentrators. Rather than many back end tools connecting directly to the front end machines, increasing the work load of these machines, they instead connect to the server. Servers like those discussed above have been used successfully at the Thomas Jefferson National Accelerator Facility to provide functionality such as beam steering, fault monitoring, storage of machine parameters, and on-line data processing. The authors discuss the potential uses of such servers, and share the results of work performed to date.

1 INTRODUCTION

The classic control system in use in many locations today consists of a collection of front end machines, distributed around a facility, largely in order to keep them close to the hardware they are controlling. These front end machines are interfaced directly to the hardware, and are responsible for maintaining variables associated with the hardware, as well as responding to queries about the state of or modifying settings for the hardware.

The users view the system from the point of view of the back end machines. These machines, often UNIX machines or PCs, run applications programs which modify the settings of hardware, display values from hardware, and monitor the control system's behavior.

Very frequently, the raw information obtained from the front end machines is not directly useful to the users, but needs to be processed in some way to make sense. Doing this in a back end program has the advantage of moving processing from the front end servers to the back end servers, which are generally less critical to the real-time control of the hardware, therefore reducing the overall load on the critical systems. This approach has several disadvantages, however.

One disadvantage is that if many programs need to look at some new value which is derived from several control system variables, each must calculate these new values independently. This increases the resource consumption on the back end servers, the chance of introducing errors into the system, and the development time for each new application that must support these new values. Additionally, these new derived values will generally not be directly available for archiving, viewing, or monitoring by traditional tools designed for direct monitoring of control system values.

2 MIDDLEWARE SERVERS

2.1 What is a middleware server?

The solution to these problems taken by the authors is to develop middleware servers. A middleware server is a program, a "software server", which obtains data from the front-end servers, calculates new values, and creates virtual control system variables for viewing by back end tools. The goal of the middleware server is for it to be virtually transparent to the user – the user should not be concerned with whether the variables are obtained directly from the front end machines or are virtual variables on a middleware server. In addition, the algorithms used to derive these new values are located in one point, and easily be modified without the need to modify the client programs.

2.2 Middleware servers at Jefferson Lab

At the Thomas Jefferson National Accelerator Facility (JLab), the front end machines are dedicated machines running WindRiver System's VxWorks, and EPICS, the Experimental Physics and Industrial Control System [1]. These systems monitor and control many aspects of the

machine, from magnets for beam optics, to beam position monitors, to cryogenics.

The back end tools at JLab are run on HP-UX UNIX machines, and consist of a mixture of the general purpose EPICS tools, such as viewers, archivers, and machine configuration save and restore tools, as well as in house developed applications. In addition, JLab uses a higher-level protocol called CDEV, for Common DEVice [2]. CDEV provides the advantage of making EPICS variables, and variables from other sources available to the user in such a way that they can be accessed with the same interface. Many of the existing tools for EPICS are being ported to CDEV, and much of the new development at JLab is based on CDEV.

CDEV is particularly well suited to developing servers, since a Generic Server engine is provided [3]. This is a simple software construct that can be used to rapidly develop middleware servers. It provides the framework for monitoring existing values from the control system, and for creating new attributes to be monitored by the back end servers. While enhancements to this framework are sometimes needed when developing an application, the developer is generally free to concentrate on developing the algorithm associated with processing the data, rather than being concerned with the framework and communications structure.

3 USES FOR SERVERS

There are may ways middleware can be used. As the authors continue to develop applications, more uses for these servers present themselves.

3.1 Servers as online data sources

One of the primary areas in which servers are useful is in providing or storing information to the control system that would otherwise either not be available, or that might be stored in front end machines needlessly. The servers can be constructed to contain CDEV variables that can be read, set, or monitored by users. These values do not need to come from the front end servers, but can be standalone values. Virtually any arbitrary value, from the names of the current operations crew to theoretical machine parameters can then be used just as if they were control system values.

Additionally, these servers can be built with logic of their own. While still not manipulating the control system, they can be loaded with theoretical values for certain parameters, and calculate new values from these, perhaps using one or more control system values in the computation. Since such servers run on inexpensive UNIX workstations, the load of performing these calculations is moved off of the front end computers.

At JLab, our Model Server Artemis is an example of such a server [4]. Two instances of this application are used. Both are initially loaded with the theoretical optics for the machine. The second instance of the model is the periodically updated with actual values for components from the machine. Based on these input values, transfer matrices, alpha and beta values etc. are calculated and made available to optics applications.

Additionally, information about the locations of signals (which front end server a particular channel resides on) is stored in such a server. This is used, along with modified versions of back end tools, to speed connection time when accessing control system channels [5].

3.2 Servers as controllers

Another useful application of the server is as an actively controlling program. In this capacity, the server functions as a less deterministic feedback system. The server monitors a number of values related to certain parameters of the control system. Based upon these values, new parameters are calculated and loaded back into the machine. This can continue periodically. In addition, the server allows the controls for the algorithm, such as parameters, whether to apply changes or not, etc., to be made available as control system signals. This makes monitoring and controlling the behavior of the server simple.

At JLab, we use such servers for several beam control applications. Three servers fall into the category of "locks". These servers monitor parameters of the beam – position within the beampipe, energy, and current – and try to "lock" them to some predefined value [6]. This is accomplished by reading the current value of the parameter one wishes to lock, calculating new values for parameters that modify the desired parameter, and applying those changes to the control system. As an example, for beam position one would read the values of BPMs (Beam Position Monitors), determine where and by how much the beam is deviating from the ideal, and apply changes to steering magnets to return the beam to the optimal location. These servers perform these checks every 1 to 5 seconds, depending on the configuration of the accelerator.

3.3 Servers as monitoring systems

Servers can also be used to provide online monitoring of values in the system for diagnostic purposes. Most control systems provide some mechanism for noticing if a single signal exceeds predefined limits and bringing this to the operators attention. Servers provide the benefit of monitoring multiple signals and inferring when a value is bad based upon its relationship with other signals.

The server can also monitor values from multiple signals, and calculate new values from these signals – a "value added" signal. As in the case of the online data sources, doing the processing at the server level saves CPU load on the front end servers, leaving them free to control hardware. It is also superior to calculating these in the client program if multiple clients need this combined information.

At JLab, this style of server is used for enhanced alarm servers, which monitor special parts of the machine and alert operators of trouble based on complex algorithms involving multiple signals. Additionally, this style of server is used in a program which calculates changes to the machines energy at a very low level, allowing interested parties to notice changes in the system.

Additionally, a new more generic form of this server is being explored. The proposed tool, called the Automator, is intended to allow for generic, user defined instances of such a server to be created and used [7]. The server could monitor for specific alarm conditions and, optionally, take predefined actions when such conditions occur.

3.4 Servers as caching devices/concentrators

Finally, servers can be used to cache or concentrate signals that are frequently accessed. By modifying the information flow so that the back end clients access the middleware server rather than the front end machine directly, the load on the front end machines is reduced, again freeing these machines for hardware control and processing. This type of server is often combined with some of the functionality of the monitoring servers mentioned above.

At JLab, we use a hybrid of this type of server and a monitoring server for BPM data. With many BPMs, and many applications interested in using them, we created a server to monitor this data. Multiple clients then connect to the middleware server, rather than connecting directly to the front end machine, which now has less connections to service.

Our server also provides several additional services. It monitors the status information provided by the BPMs and produces enhanced status information. It filters out transient failures in the BPMs, and attempts to ensure that different attributes of information about a given BPM are correlated in time. This ensures that the client sees an accurate picture of the machine status.

4 EXPERIENCE

The experience the authors have had with these servers has generally been positive. These servers seem to provide a reliable, simple way of implementing what would otherwise be very complex actions. There have been problems associated with these servers, as there are with all software applications, and perhaps a tendency to use the tool to try to solve all problems, but the concept seems to be very sound.

The authors have also found that these servers have practical limits to how large or how rapidly they can process information. Since these servers are monitoring values, and posting monitors to clients on changes, one must be careful not to overburden the code. One server developed on site attempted to process one hundred million events (changes in the control system that required modification to virtual variables) per day, or about 1000 events per second. The peak load during transient events (such as beam turning on or off) was much higher than 1000 events per second. This server exhibited occasional problems with coherence with the control system, particularly immediately following high event count peaks. The solution was to split this server into small servers, to reduce the high number of events handled.

Similarly, the active feed back programs, or locks, have a limit on how fast they can process. This is partially determined by the algorithm and the time needed to calculate a solution, but is also limited by the time needed to monitor the signals from the control system. For numerically intensive calculations a feedback loop of approximately 1Hz. seems to be a comfortable top speed for such servers, when running on a Hewlett Packard K-class machine.

5 CONCLUSION

In conclusion, these servers offer the developer a powerful tool for enhancing capability, often without the need to further burden front end servers or modify tested, working front end code. It is not a panacea for every control system problem, but when used properly is a powerful and effective way of addressing certain software problems.

6 REFERENCES

[1] http://www.aps.anl.gov/asd/controls/epics/EpicsDocumentation/ EpicsGeneral/epics_overview.html

[2] J. Chen, G. Heyes, W. Akers, D. Wu and W. Watson III, "CDEV: An Object-Oriented Class Library for Developing Device Control Applications", Proceedings of ICALEPCS 1995

[3] W. Akers, "An Object-Oriented Framework for Client/Server Applications", Proceedings of ICALEPCS 1997

[4] B. A. Bowling, W. Akers, H. Shoaee, W. Watson, J. van Zeijts, S. Witherspoon, "Evaluation of a Server Client Architecture for Accelerator Modeling and Simulation", Proceedings of CAP 1996

[5] D. Jun, D. Bryan, W. Watson, "Centrally Managed Name Resolution Schemes for EPICS", Proceedings of ICALEPCS 1997

[6] J. van Zeijts, et al., "Design And Implementation Of A Slow Orbit Control Package At Thomas Jefferson National Accelerator Facility", Proceedings of PAC 1997

[7] D. Bryan, M. Bickley, K. White, "The Automator : Intelligent Control System Monitoring", these proceedings (1999)