Parallel Virtual Machine (PVM) ready for distributed real-time control systems?

Case study for industrial usage.

Armin STEINHOFF
STEINHOFF Automation- & Fieldbus-Systems
D-65604 Elz / Germany armin@steinhoff.de

ABSTRACT

For distributed control systems is often used Industrial Ethernet, but the standardization of the upper layer protocols is still an open issue. The use of PVM is an interesting approach to build a standard at application layer.

With PVM it's also possible to integrate to a real-time environment additional existing programming standards like IEC 61131-3 as well as other requested features from the control industry.

PVM is a quasi standard for parallel computing systems and it has a big and still growing installed base and is used at many sites, but it's not common to use it in distributed real-time control systems today.

Our first trial with PVM in a distributed real-time control systems looks very promising.

Keywords:
PVM, Industrial Ethernet, distributed control systems, message passing, fieldbus integration, programming in Python, C, IEC 61131-3 etc.

CURRENT DISTRIBUTED CONTROL SYSTEMS

Ethernet is used since years for distributed control systems. The current discussion about the industrial Ethernet includes also issues about the upper layer protocols and its standardization. This discussion happens in several autonomous groups of interests doing all their own definitions of the upper layer stacks. All of these efforts seem to be started from the scratch or trying to introduce proprietary solutions from different vendors.

Until now there are no promising approaches visible for a common accepted communication standard for industrial distributed control systems.

Could PVM [1] be a base for an alternative solution?

PVM FOR DISTRIBUTED CONTROL SYSTEMS

From a communication and processing point of view, distributed control systems are very similar to loosely coupled heterogenous parallel computing systems, even if they have to fulfill different processing tasks and different requirements.

Fact is that a lot of mature software solutions for programming and controlling of parallel computing systems are already available. Parallel Virtual Machine (PVM) and several MPI based systems like e.g. LAM-MPI [2] a.s.o. are well known today but usually not used for industrial control applications.

These parallel computing software systems are in general free open source systems. Since years they are widespread used and are supported by a huge number of science and research sites and different industrial users. Most of these loosely coupled parallel computing systems are using message passing communication. It's based on the lower layers of TCP/IP.

PVM library on top of PROFIBUS

It is important to know that the PVM library definitions are not bound to a specific transmission media or network technology, that means the real-time behavior and determinism of a PVM based system is not an issue of the library definition itself, but it depends on the used network media.
For that reason it's possible to get hard deterministic real-time behavior by implementing the PVM library on top of a deterministic fieldbus like PROFIBUS-DP (DPV1, FDL at 12Mb/s) [3] or other standard fieldbus systems as well as on other deterministic transport media.

**Important PVM features for control systems**
PVM provides process and resource control functions and offers also services for fault detection at task level.

The support of different language bindings and the included system transparent programming model are very important features for the usage of PVM in heterogenous industrial distributed control sysytems.

**Benefits of PVM for distributed control systems:**
PVM is much more than industrial Ethernet plus a TCP/IP based communication stack.

- PVM provides a programming standard at application level accross system borders.
- PVM offers interoperability for heterogenous computing environments. That means e.g. Windows and Unix Systems are able to communicate flawless together.
- PVM provides management services for its virtual machine.
- PVM is reconfigurable at runtime.

After all these considerations PVM looks like an ideal open solution for distributed control systems.

**SOLVED ISSUES FOR INTEGRATION**
A remaining question for making a PVM system usable for industrial requirements is how to integrate existing programming standards like IEC 61131-3 [4] in that PVM environment, especially when we have to deal with real-time conditions.

**IEC 61131-3 Programming Standard**
This programming standard is widely used in the world of control applications. It defines two programming languages and three programming frameworks which all are referencing a common process data base.

**IEC 61131-3 Kernels**
The execution environment of IEC 61131-3 coded applications consists of a so called kernel. That kernel executes the uploaded application code of the control application. At upload and debug time exist a communication link between the kernel and the graphical programming workbench.

High level communication functions are not covered by the IEC 61131-3 processing model. Such functions must be realized externally by C-coded function blocks, which become an integral part of the execution environment.

**Integration of PVM functions into the IEC 61131-3 Kernel**
The integration of PVM functions into the kernel code increases the size of the kernel in a big amount and imports also the possibility for software faults. A crash in a C-function block can shutdown the complete kernel.

The code size problem can be solved by dynamic loadable shared libraries, but it doesn't solve the negative impacts of software faults.

This aspect of software faults seems only to be solvable by a class of operating systems with fast message passing based interprocess communication.

A possible solution for the kernel extension is the usage of client/server processing based on message passing. It allows the realization of functional extensions for PVM-API services in a server process without changing the code at the client's site in a remarkable amount.

Each client can request services from a server process if such an extended function is needed. In our case it requires a very fast message passing implementation, which is available by e.g. QNX [5] with message cycles in a range of 4us on a 400Mhz machine. QNX supports also separated protected address spaces for the client and server processes, what is the key for software fault tolerance.

We consider the IEC 61131-3 kernel in that solution as a client, which requests non-IEC 61131-3 services from a specialized server by using message passing. It's important to know that such a client must be changed only once for the implementation of the message passing communication, which leads to small and stable IEC 61131-3 kernel implementations. That's just what we need for a reliable solution.

PVM gives us additional the freedom to realize language bindings for optional programming languages which allow to implement a PVM-API server in plain C, Python [6] or any other by C-Code extensible Interpreter language like ICI [7], LUA [8], Perl [9] or TCL [10].

The prerequisit for an implementation in Python is e.g. only a QNX related message passing extension of Python. With this it is possible to create a PVM-API server on the fly with Python using the PyPVM extension.
Extension of the IEC 61131-3 workbench
In addition to the PVM-API server there is also to implement a function block library for the IEC 61131-3 programming workbench.

Each of these function blocks are representing the respective call of the PVM API and provide via message passing the call parameters.

The appropriate code at the server site realizes just the call into the PVM API.

This can be done with convenient graphical frame works like SFC (sequential function chart), FBD (function block display) or the textual language ST (structured text).

Internet
At the end Internet can also be included into that distributed environment, so we will be able to write applications running even around the globe :).

Xpwm & graphical configuration tools for PVM
Xpwm and available graphical configuration tools for PVM are additional possibilities in our consideration, so it's to say that the usage of PVM for distributed control systems is very promising.

PVM Data Base, Central Visualization and Fieldbuses
An interesting feature of PVM is the simple data base of PVM. It allows to store and retrieve data referenced by name and index.

It is useful for an application like a central visualization which needs access to all distributed data

BENEFITS OF THE IEC 61131-3 INTEGRATION

PVM function block library
The PVM function block library for the workbench enables to program distributed control applications without leaving the IEC 61131-3 programming environment because of dealing just with one single virtual machine.
comming from e.g. distributed I/O images handled by fieldbus drivers. These data can be stored with a definable cycle by the respective driver process into the PVM data base and can be used for visualization purposes.

**SUMMARY**

The current version of PVM offers a lot of very useful services for distributed control systems. It's available with all sources and can easily be ported to other computing systems. It can also be adapted to specific requirements.

It's time to use PVM in industrial control systems to have a common communication standard which can integrate flawless all needed industrial standards and "proprietary standards" from different vendors.

With integration of fieldbuses and all different existing "Ethernet standards" as well as programming standards PVM looks like having a great future for industrial usage.

Our first prototype with PROFIBUS-DP, IEC 61131-3, QNX6, Ethernet, MMI and Internet access looks very promising.

**REFERENCES**

[1] PVM  
http://www.epm.ornl.gov/pvm/pvm_home.html  
http://sourceforge.net/projects/pyqnx

http://www.lam-mpi.org

http://www.profibus.org

http://www.plcopen.org

[5] QNX: RTOS, QNX is reg. trademark of QSSL  
http://www.qnx.com

[6] Python: OO Interpreter  
http://www.python.org

[7] ICI: C-Interpreter  

[8] Lua: OO Interpreter  
http://www.tecgraf.puc-ria.br/luau

[9] Perl: Interpreter  
http://www.perl.com

[10] Tcl: Interpreter  
http://www.scriptic.com