PORTING PVM TO A SHARED-MEMORY INDUSTRIAL MULTIPROCESSOR

E. Appiani, M. Bologna, M. Corvi
Elsag Bailey, *Research & Development Services, v. Puccini 2, 16154 Genova, Italy*

M. Ancona, G. Dodero, V. Gianuzzi, M. Iardella
Univ. of Genova, *DISI (Dept. of Computer and Information Science), viale Benedetto XV 3, 16132 Genova, Italy*

A. Clematis
CNR-IMA *(Inst. of Applied Mathematics of Italian National Research Council), Torre di Francia, v. De Marini 6, 16149 Genova, Italy*

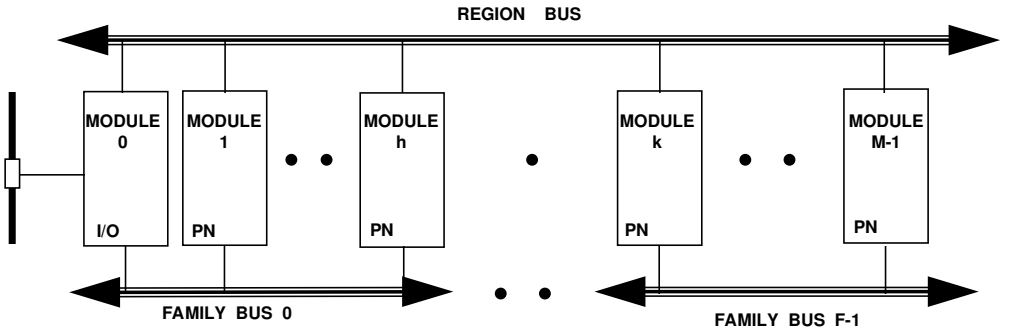
Abstract - The paper presents the experience of porting the PVM message-passing environment to EMMA2E, an industrial, shared-memory multiprocessor architecture. The work is still ongoing in the framework of the ESPRIT "EUROPORT2" project, aiming to demonstrate practical availability of portable parallel applications with related industrial and commercial benefits. The project framework, the EMMA2E architecture and the related system software are briefly introduced, followed by the main features of PVM porting solution. This last especially aims to join PVM general and portable programming model with the run-time efficiency achievable in a shared-memory environment.

1. EUROPORT2 and PULSAR

EUROPORT2 is a project in the ESPRIT III framework, aiming to develop portable parallel applications in order to demonstrate that parallel computing may be employed in industrial and commercial environments on a large scale. The key to win users' fears about the variety and complexity of parallel architectures and tools is *portability,* by which a program can be parallelized on a machine-independent model and then hosted on different architectures. *Message-passing environments* meet such requirements.
EUROPORT2 is divided into international consortia, each one identifying an *application software package* and a *message-passing tool* for parallelization and portability. Elsag Bailey participates to the PULSAR consortium [6] with French (Scot Conseil) and English (Univ. of Liverpool and NA Software) partners, also cooperating with the University of Genova and the Italian National Research Council (under support of the strategic project "Mathematics for Technology and Society"). The application is a package for *advanced SAR image processing* and the message-passing tool is *PVM* [4] [2].
Final PULSAR deliverables are due out by Dec 1995; the main activities of the first year, still in progress, are the package parallelization on PVM, the definition of test and benchmarks for performance evaluation, and porting PVM to EMMA2E.

2. EMMA2E architecture and system software

EMMA2E is the most recent machine in a family of industrial parallel computers employed by Elsag Bailey since the late 70's, for computing-intensive pattern-recognition, image and signal processing applications. The architecture is a *shared-memory scalable collection of processing modules, connected by a double hierarchy of linear busses* with 64-bit data path and 32-bit address path, sketched in fig. 1. Processing modules may be: *I/O front-ends,* handling standard or custom peripherals with one processor each; and *computing modules,* also called PNs, with 4 processors each. A processor is based on a 486 running at 33 MHz, possibly coupled with a DSP96001 as *slave coprocessor* for intensive floating-point computation, sharing the same address space of the main processor.
The maximum configuration is made of a LAN interface module, needed for communication and supervision from a standard host node, and 20 PNs with 80 processor-coprocessor sets, with a total peak computation rate of about 1000 MIPS + 2000 MFLOPS. The peak rate in a bus is 64 MB/s, when moving data between shared memories with custom programmable DMA controllers. The total rate depends on the number of secondary busses in a configuration; in the most typical cases there are one primary bus and 3 secondary ones, giving 256 MB/s peak rate.



EXAMPLE SCHEME OF AN EMMA2E TARGET NODE WITH

- **ONE** I/O MODULE FOR LAN INTERFACE
- **M-1** PN MODULES, FOR A TOTAL OF **M** MODULES AND **4M - 3** PROCESSORS
- **F** FAMILY BUSSES

- Fig. 1 -
EMMA2E architecture

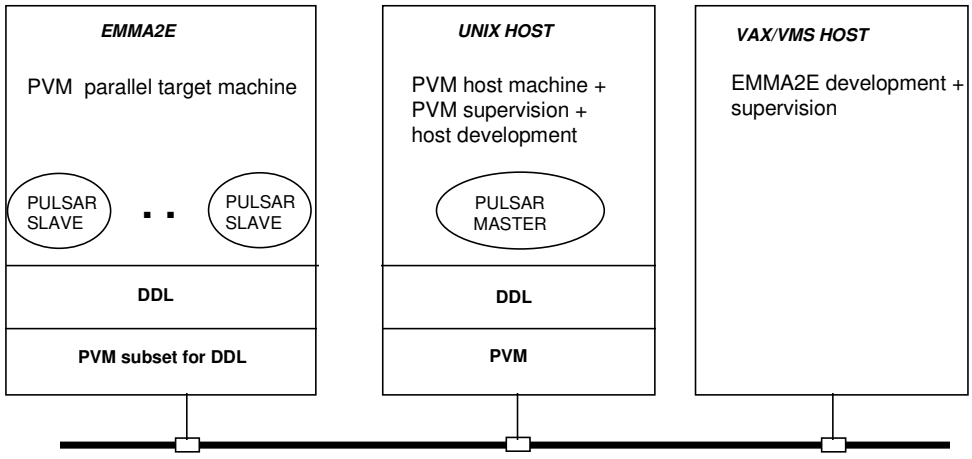
EMMA2E system software is EROS, a proprietary operating system offering interactive and run-time services [8].
Interactive services are used for machine configuration, control and parallel program development, including a symbolic debugger with multi-tasking interface. They run on any VAX/VMS host, also acting as standard I/O server, linked to the target node by Ethernet, UDP/IP network.
Run-time services offer a wide range of primitives for kernel and I/O services, including support to parallel computation (shared memory, mutual exclusion, inter-processor synchronization), remote file I/O and custom device management (shared-memory DMA, parallel I/O and coprocessors).
EROS allows for accurate development and fine tuning of industrial parallel applications, but its proprietary interface limits EMMA2E to usage inside the company and shipment to clients only within integrated, turn-key systems. PVM porting is an interesting experiment for combining EMMA2E special computation features with a de-facto standard software environment.

3. Software architecture of PVM for EMMA2E

PVM requirements for PULSAR do not extend to the whole library, since the parallelization is based on an intermediate library for array and matrix operations on distributed data, called DDL (Data Distribution Library) [6]. The parallel model is *master-slave:* the application master task is the user front-end on a Unix host while the collection of slave tasks perform the data parallel computation on the target machine. PVM supports this model very well, as shown in [2] and [3].

The minimal configuration required for porting PVM is a *Unix node with Ethernet link to EMMA2E.* A further VAX/VMS or AXP/OpenVMS host, as shown in fig. 2, does not participate to PVM operations, but is needed to start and control the target EMMA2E machine. PVM primitives to be ported on the parallel machine for DDL are just 13 of the 51 included in PVM 3.x [1]: *pvm_mytid(); pvm_exit(); pvm_initsend(); pvm_bufinfo(); pvm_getrbuf(); pvm_{pk/upk}{char/int}(); pvm_send(); pvm_mcast(); pvm_recv();* and *pvm_probe()*. They are few but provide the true core services for parallel and distributed computation.

Despite PVM porting is currently ongoing, we think however that most PVM primitives will be implemented instead of just the ones mentioned above, especially due to the Daemon implementation illustrated later.



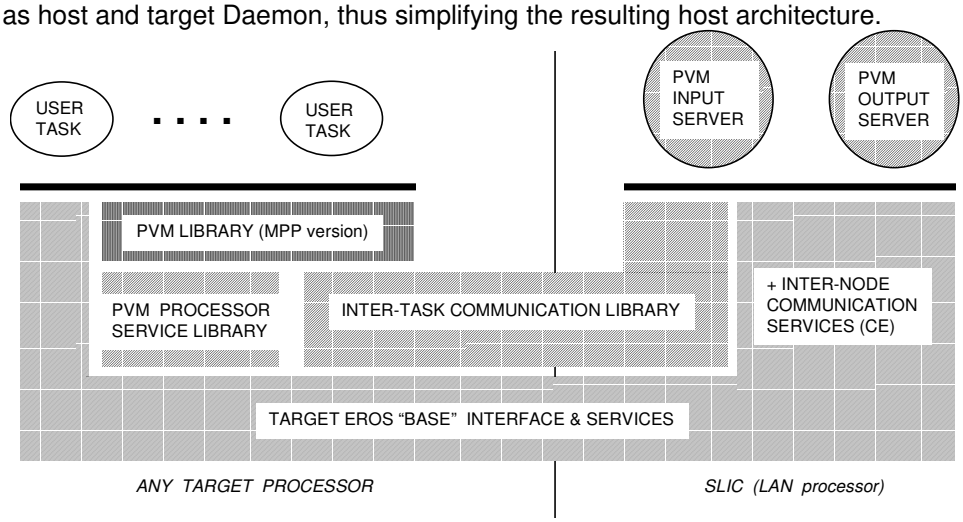
- Fig. 2 -
Minimal distributed configuration for PULSAR under PVM on EMMA2E

So far, PVM has been ported to several commercial multiprocessors [3] [4] [5], mainly distributed memory ones. Basic requirements for multiprocessors, suggested in PVM manual [1] and confirmed by examining the source code, are the following:

- **PVM Library:** running on any processor; interfacing with kernel services mostly for memory management; and interfacing with a custom library relying on machine-specific *inter-processor communication facilites,* for direct communication among tasks located on the same multiprocessor;
- **PVM Daemon:** running on the LAN interface processor or on the host front-end; interfacing with Unix and standard I/O services; communicating with other Daemons by means of Socket-UDP; and, finally, interfacing with a *custom target interface* relying on specific *inter-processor and/or inter-node communication + task loading facilities,* depending on Daemon location.

The first problem to solve was the *Daemon location*. It could run on EMMA2E LAN interface processor, with EROS interface + standard communication (Socket - UDP), or *on the Unix host*. The second choice appeared to be more convenient, since the multiprocessor version of the Daemon differs from the standard one just in communication with tasks of the target node, otherwise relying on Unix services, whose porting to EROS would have exceeded the reasonable effort planned for this project, possibly limiting Daemon services.

Putting the Daemon on the Unix host just requires implementation of the custom target interface (see above) and of two message-routing tasks on the LAN processor, as Daemon front-ends. Besides, the same task may act both as host and target Daemon, thus simplifying the resulting host architecture.

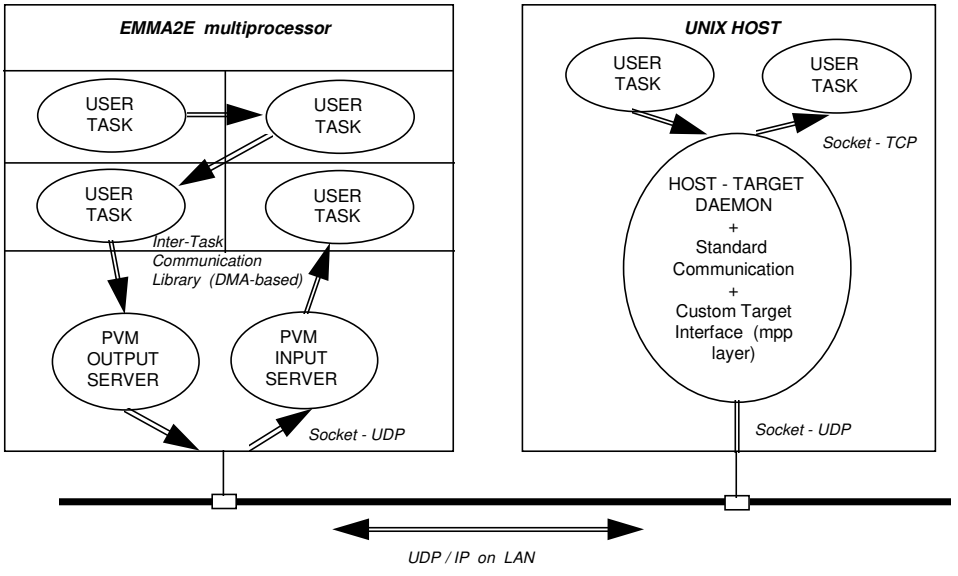


- Fig. 3 -
PVM architecture on EMMA2E multiprocessor

The PVM architecture for EMMA2E multiprocessor node is shown in fig. 3. It includes the following components, implemented upon EROS operating system:

- **PVM Processor Service Library,** an EROS-based layer mainly implementing the C standard interface for memory management. On EMMA2E this is based on a *heap in shared memory,* for achieving word-granular allocation/deallocation (EROS manages memory space on a 4KB page basis) and multiprocessor sharing of PVM buffers;
- **Inter-Task Communication Library,** a simple interface for exchanging messages between tasks located on different processors. Message data are directly moved from source buffer to destination one by employing the custom DMA controllers. This messaging facility is the core of efficient multiprocessor communication, whose approach is better illustrated in the next paragraph;
- **PVM Input Server,** a task that waits messages from the host Daemon for EMMA2E via Socket - UDP and routes them to the target tasks via Inter-Task Communication. Target tasks are addressed by using their own processor identifiers in the corresponding PVM task identifiers (tid), found in the message headers;
- **PVM Output Server,** that reversely waits messages from target tasks via Inter-Task Communication and routes them to the host Daemon via Socket - UDP. This couple of tasks results in simpler dynamics for each one and in full-duplex host-target communication, also thanks to EROS multitasking facilities on each processor.

On the Unix host, the Daemon for EMMA2E is implemented as a standard process, similar to any other Daemon but relying on custom communication on the host-target side. The host may run both the local and target Daemons, thus leading, in a two-node PVM configuration, to the communication patterns and related means summarized in fig. 4.



- Fig. 4 -
PVM communication on host-target configuration

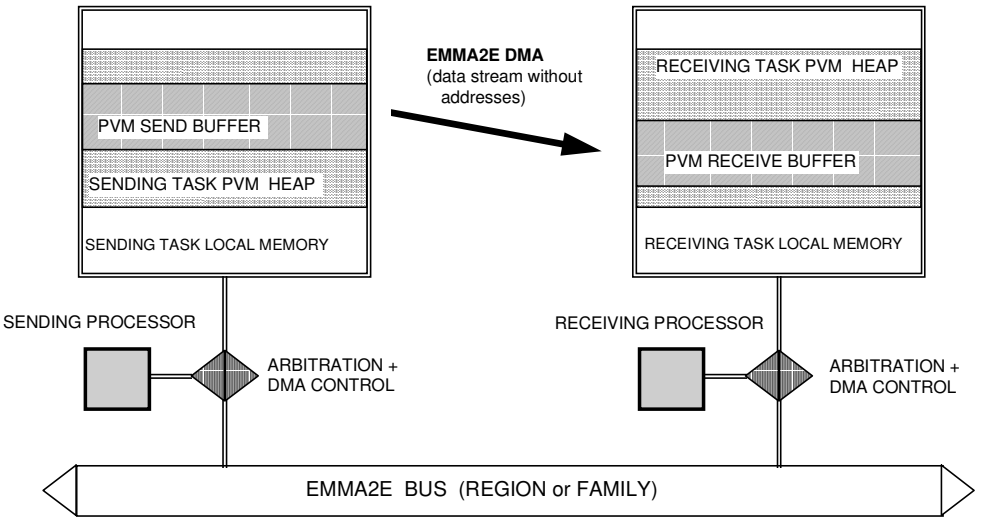
4. Implementation approach of Inter-Task Communication

Direct message exchange on EMMA2E, based on shared memory and fast data moving, was studied and experimented in [7]. The result was a library of *synchronous inter-task message passing,* by which sender and receiver tasks have to synchronize with each other in order to have their respective *data buffers available at the same time.* Being both in shared memory, it is possible to *directly move data between them* through the custom DMA controllers (see initial introduction to EMMA2E) and achieve the maximum bus data transfer rate. Of course, the average message rate is slower, due to mutual synchronization overheads.

For implementing such scheme, memory "heaps" for allocating and deallocating PVM buffers are reserved by each task, at the moment of joining PVM (pvm_mytid), in the local shared memory areas of corresponding processors, as shown in fig. 5. Pvm_pack and pvm_unpack operations are performed by PVM tasks internally to their processing modules, thus not using global busses and being parallel among processors.

The core of message transfers resides in the DMA between buffers in different shared memories, so that data cross just one bus for each message exchange.

Under PVM, this library was extended to include services of *asynchronous send, test for completion* of previous asynchronous sends and *conditional receive.* These operations give best support to pvm_send(), since it internally schedules asynchronous operations on multiple buffers and keeps track of all busy ones, by polling for completion of related pending operations.



- Fig. 5 -
DMA as core of PVM message exchanges

5. Current status and future work

At present, all the presented services for PVM on EMMA2E and on the Unix host are under separate testing. Integration and PVM testing on the host-target configuration are planned to finish by the end of October. Afterwards, PVM performance will be measured in order to find possible bottlenecks and services to be improved. During next year, PVM for EMMA2E will be integrated with the DDL library (implemented at Univ. of Liverpool, UK) and, later, with the PULSAR parallel package (parallelization is going on at Univ.of Liverpool and NA Software Ltd., UK). PULSAR will be finally evaluated against performance and result correctness on EMMA2E and other parallel/distributed machines (Meiko CS-1 and a network of Sun workstations).
Further research activities on PVM model are planned. The perfomance study of EMMA2E DMA-based message passing, described in [7], will be adapted to model the PVM library as well. The next merging with another methodology for modeling pipelined message-passing applications, described in [9], will possibly result in a tool for predicting the performance of pipelined applications using PVM on EMMA2E or other machines. Such tool might be the key for successful parallel implementation or porting of industrial applications.

Bibliography

[1] Geist, Beguelin, Dongarra, Jiang, Manchek, Sunderam, "PVM3 USER'S GUIDE AND REFERENCE MANUAL", ORNL, Oak Ridge, TN 37831, USA, May 1993

[2] V. Sunderam, "PVM: A FRAMEWORK FOR PARALLEL DISTRIBUTED COMPUTING", paper from research performed at ORNL, Oak Ridge, TN 37831, USA, 1992

[3] V. Sunderam, "HIGH-PERFORMANCE CONCURRENT COMPUTING WITH PVM" (PVM training), Emory Univ., Atlanta, GA (USA), Apr 1994

[4] U. Block, W. Gentzsch, "PVM FOR EUROPORT", presentation at GMD (St. Augustine, Germany) for EUROPORT kick-off, Jan 1994

[5] W. Gentzsch, U. Harms (editors), "HIGH PERFORMANCE COMPUTING AND NETWORKING", proceedings of Intern. Conference and Exibition (Munich), Springer-Verlag, Apr 1994

[6] ESPRIT EUROPORT2 TECHNICAL ANNEX: PART H, "PARALLEL UNIVERSAL LIBRARY FOR SAR", Smith Engineering, Sep 1993

[7] M. Iardella, "SCAMBIO MESSAGGI SU ARCHITETTURE MULTIPROCESSORE: ASPETTI IMPLEMENTATIVI E PRESTAZIONALI", Italian Thesys for B.S. Degree in Information Science, Univ. of Genova, Apr 1994

[8] E. Appiani, "EMMA2E EROS: INTRODUCTION AND GENERALITY", Elsag Bailey doc. n. RJ4.9034.132.60, Nov 1993

[9] P. Kenyon, S. Seth, P. Agrawal, A. Clematis, G. Dodero, V. Giannuzzi, "PROGRAMMING PIPELINED CAD APPLICATIONS ON MESSAGE PASSING ARCHITECTURES", proceedings EWPC '92: PARALLEL COMPUTING FROM THEORY TO SOUND PRACTICE, IOS Press, 1992, pp. 550-553