Open HPEC Systems: Design and Profiling tools for multiprocessor signal processing applications using MPI

Benoît Guillon, Jérôme Blanc, Benoît Masson, Gerard Cristau, Vincent Chuffart
Thales Research and Technologies, Thales Computers

Early benefits of adopting a standard for multiprocessor communications include the opportunity to use existing tools, or adapt them very easily.

Morphing them into design tools supporting the specific needs of high performance embedded applications has a return of invest which is not limited to one particular hardware platform.

Using MPI for HPEC applications: Zero Copy Communications

As the first step, we have developed a high performance implementation of MPI enabling "zero copy" communication protocols on high-speed networks (i.e., hardware DMA on Fiber Channel, RapidIO, or shared memory) in addition to the standard TCP/IP protocols.

However, this feature alone is not enough as a key difference between scientific multiprocessor software and embedded signal processing applications is the need for full control over the deployment of the application: allocation of processors to processes, selection of physical communications channels, and the amount of buffering occurring between computational stages.

Augmenting MPI: Topology files

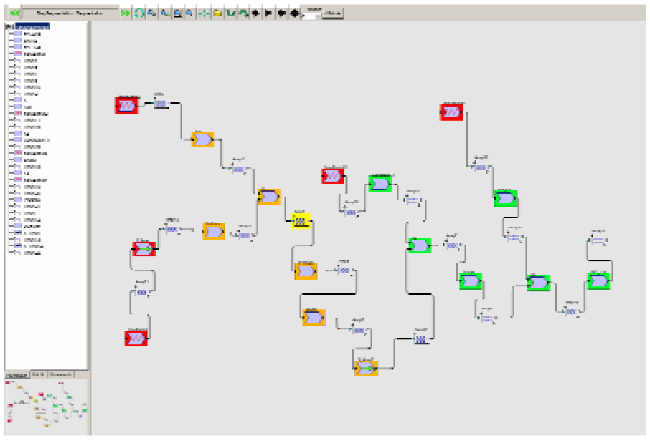
All the specific HPEC application requirements described above are better placed under the designer control, rather than buried amongst the application algorithmic parts, embedded in the operating system or hidden in some middleware layer.

Thus was also implemented the concept of topology files which contain all the deployment parameters for a given HPEC application, such as process locations, complete description of the I/O channels, (like physical media and buffering information). This work was presented during HPEC2004 (poster C5).

This level of control contributes to guarantee the determinism of run-time behavior, which is always desired for application test and debug, and often required when the signal processing application is part of a critical system.

Application design tool and graphical interface

The next step is to manage this information in an intuitive way, using a graphical view of the application structure and its deployment strategy. Using a formalism that allows application modeling and mapping, our implementation enables the designer to keep a high-level view of the application deployment, and to focus on signal processing algorithms implementation - the communication code and deployment configuration files (topology files) being automatically generated by the tool.



High-level view of a HPEC application

We have experimented the use of Ptolemy II, the design environment of the University of California at Berkeley [1], as a graphic interface for high performance signal processing applications based on MPI.

Feedback on mapping strategies: MPI trace analysis

As described in [3], a pragmatic approach to HPEC application mapping involves a man-in-the-loop scheme, where various mapping strategies on a given architecture can be experimented. This requires feedback information on the run-time behavior of such strategies.

We present a completely standalone system, which includes multi-processor PowerPC based VME boards and different communication channels between processors along with a Pentium based VME board which runs the tools on top of the Linux Operating System.

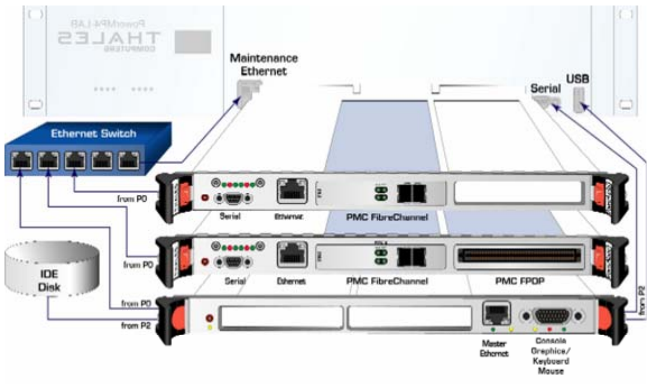


Figure 1: Demo System.

We use standard MPI profiling utilities (the MPE libraries, and the Jumpshot graphic tool from the University of Indiana [2]) to analyze the run-time behavior of signal processing applications.

The following images present the same application, modeled with the help of the graphical tool, after two different runs, done with different I/O mapping strategies, to illustrate the impact on run-time behavior.

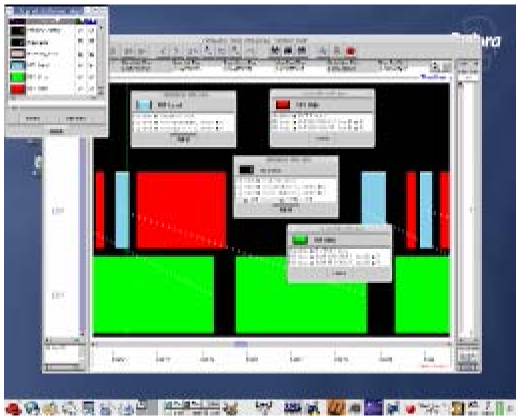


Figure 2: Jumpshot view of MPI traces (VME2eSST)

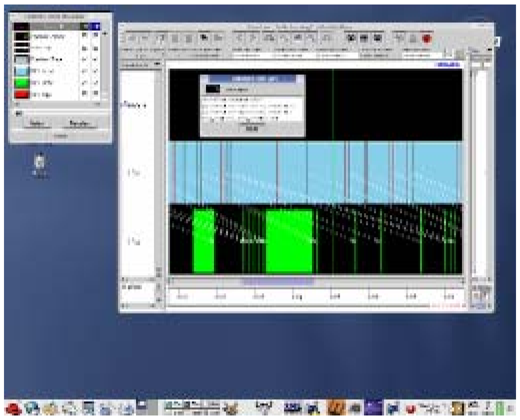


Figure 3: Jumpshot view of MPI traces (TCP/IP)

Other typical examples will be run on the system during HPEC2005.

Conclusion and future work

Our pragmatic approach strives to use state of the art technologies for implementation: within the Ptolemy framework, we are using a "model based" approach, i.e., rely on a formal internal representation of: the application, the hardware architecture, and the mapping of the application to the architecture. Preliminary studies [4], show the feasibility and benefits of simulations based on the design model of high performance signal processing applications.

Besides, standardization efforts (such as [5]) are addressing the needs for HPEC design models which are both formal, and able to take benefit from the expected availability of a wide range of UML 2.0 and MDA tools.

Such efforts do not necessarily cover all aspects needed in signal processing applications [3]. For example, interoperability between different modeling techniques, addressed by Ptolemy, is not a major concern of UML tools today.

However, we think the same rules apply for design tools as for run-time libraries: adopting standards has a return of invest through the possible reuse, or easy adaptation of existing tools.

References

[1] Edward A. Lee "Overview of the Ptolemy Project", *Technical Memorandum UCB/ERL* M03/25, July 2, 2003

[2] Anthony Chan, David Ashton, Rusty Lusk, William Gropp4 "Jumpshot-4 Users Guide" Mathematics and Computer Science Division, Argonne National Laboratory

[3] E. Lenormand, G. Edelin "An Industrial Perspective: Pragmatic high-end signal processing environment at Thales", *3rd international workshop on synthesis, architectures, modeling and simulation*, Samos, 2003.

[4] C. Dumoulin, J.-L. Dekeyser, B. Kokoszko, S. Pulon, G. Cristau "Interoperability between design and simulation tools using model transformation techniques", *FDL'03*, Frankfurt, September 2003

[5] Carnegie Mellon Software Engineering Institute "SAE Architecture Analysis & Design Language (AADL)", http://www.aadl.info

[6] Data Reorganisation Interface Standard, http://www.data-re.org