Delivering on Standards: Balancing Portability and Performance

John Robinson

Mercury Computer Systems, Inc.

Abstract

As high-performance embedded computing systems become more commonplace in a variety of applications, the need for supporting standards becomes more critical. Specifications developed by concensus, such as Message Passing Interface (MPI) and Vector, Signal, and Image Processing (VSIP), and "de facto" standards such as MATLAB® , provide a means for developers to create real-time applications across multiple platform technologies. The balance between portability and performance presents some significant challenges, including balancing the application of tools tuned to specific platforms with the use of standard, but possibly slower, code and tools.

1. Standards in Software Development

Wherever products become commodities, standards are involved. Just as a few programming languages – standardized across platforms – have become the foundation for most business applications, the commoditization of embedded computing systems requires standards to provide application portability.

Embedded hardware already adheres to many standards, from VME at the system level, down to DIN standards for connectors and VHDL for designing new hardware. These standards and many others speed the design and development of hardware products, especially in complex parallel processing systems. Yet the speed and cost of deploying software for new generations often limits how quickly developers can take advantage of new generations, and thereby rapidly insert new technology into systems such as medical imaging and government signal processing products. Likewise, the time needed to develop and deploy an application in the first place virtually guarantees that its original platform will be many generations obsolete by the time the code is finished.

Standards, including common toolkits and common (or at least, compatible) programming methodologies, can accelerate the software development and porting process for embedded systems, just as it has speeded those tasks for desktop and mainframe applications. Newly emerging tools and standards provide the foundation for breakthroughs in application development, while simultaneously raising a new set of issues and considerations for the developer to balance.

2. Emerging Tools for Software Portability

Both concensus-driven and de-facto standards are emerging in the software development realm, bringing with them the potential to increase portability, maintainability, and rapid deployment of new technology for high-performance applications. Key areas include interprocessor communications in real-time multicomputers, and common libraries for developing software and optimizing performance.

2.1 Message Passing Interface

Coordinated computation of multiple tasks in multiprocessor computing requires a protocol for scalable interprocessor communications. Message Passing Interface (MPI) has emerged as a defacto standard API for this task. MPI Software Technology, Inc. (MSTI) offers a commercial implementation of MPI designed specifically for Mercury's RACE® computing environment. MPI is in use worldwide on a variety of platforms, with broad acceptance and a fast growing application base. Supported platforms range from high-performance computing (HPC) environments to clusters of desktop workstations. It is easy to use: a full application can be developed using only six functions, while more complex applications can exploit a number of advanced features of the specification. Many applications already exist in MPI, and the specification has worldwide utilization and acceptance.

MPI/Pro for RACE supports Mercury's MC/OS® development and runtime environment. MPI-based applications can be moved between platforms that support MPI. This provides a transportable message-passing layer for portable software applications.

2.2 VSIP and MATLAB

The emerging Vector, Signal, and Image Processing (VSIP) standard, currently under development by the VSIP Forum, promises to let programmers move signal and image processing applications between computer platforms more quickly and easily by eliminating the need to recode common mathematical operations or function calls.

VSIP will provide a consistent application programming interface and common set of mathematical functions for vector signal and image processing on embedded, real-time computer systems. Mercury plans to implement the newly defined VSIP Lite subset. VSIP Lite supports specific signal processing calls that are common to Mercury's key markets and applications.

Similar to VSIP is The MathWorks' MATLAB language, which allows developers and programmers to use common MATLAB® calls in place of custom-coded functions. The RACE MATLAB Math Library lets developers rapidly target M-file programs developed in The MathWorks' MATLAB language to Mercury's RACE Series systems. The integrated environment of the RACE MATLAB Math Library allows users to compile MATLAB M-files to produce executable C code for rapid prototyping, eliminating the need to hand code complex mathematical functions in C. Primarily a research and development tool, this system helps users more accurately size applications and identify how much computer power applications will require. It is possible, however, to use code created by this system as part of the deployed application.

3.0 Clash of Standards and Performance

Both the RACE MATLAB Math Library and VSIP illustrate standards, de-facto or actual. Many "standard" software tools provide a common interface to a tool set, which can then be compiled or otherwise deployed automatically to multiple hardware platforms. High-level programming languages are a good example: C is C is C (or should be), whether it runs on VAX 9000, a Dell desktop, or an embedded PowerPC.

But standard code is not always fast code. Performance gains are almost always possible through careful, skilled optimization of instructions for particular processors and hardware configurations. This includes tasks such as memory management and buffer utilization that high-level programmers, almost by definition, don't worry about, but which consume the workdays of performance-hungry embedded systems developers. They can shave cycles, seconds, or even more from the execution time of complex tasks, but only on a particular set of hardware. Forget about moving the same code to next year's new processor architecture without major revisions if not total redeployment.

In fact, unless code compatibility is a serious issue for hardware manufacturers, hand-tuned code may not even move smoothly to the next generation of this year's processor. According to Moore's law, processor performance can be expected to double every six to 18 months. By extension, if you tune an application with nonstandard code for 25% performance gains today, next year's doubling of performance in processor power alone could actually leave your design far behind the performance curve.

Beyond "standards," there is a subset of tools which often provides much of the same benefits, with higher performance, and less risk. Proprietary runtime libraries are a good example. Sets of functions hand-tuned to a particular processor or environment generally yield the highest practical performance available on that group of processors. The downside to using such tools is that code developed using these libraries runs only on the supported hardware.

On the other hand, the very significant upside of libraries such as these is that their developers usually warrant to keep them compatible with future generations of their supported products. This means someone else – hopefully, a trusted supplier – assumes responsibility for providing new technology quickly, with compatibility for your hand-tuned code.

Table 1. Comparison of Function Performance

Function	MATLAB	VSIP (projected)	SAL
Dot product (real)	0.101	0.019	0.017
Scalar multiply	0.062	0.017	0.015
Complex FFT	0.929	0.450	0.408

Note: Performance shown in milliseconds for MATLAB and SAL was measured for 1024-length vectors on a RACE PowerPC 603 @ 200 MHz. VSIP performance is projected based on Mercury estimates. Faster processors, such as Mercury's implementation of the PowerPC 750 @ 375 MHz, will perform significantly faster.

4.0 Optimizing Versus Enduring

What is the difference between performance of standard-based and hand-tuned code, and how can developers invest their time appropriately between the two?

Comparing RACE MATLAB Math Library code performance to Mercury's Scientific Algorithm Library (SAL), a set of runtime functions optimized to run on Mercury-supported processors, shows performance penalties of up to 90% on some functions such as certain FFTs (see Table 1). This costs almost as much in performance today as you would expect to gain in 18 months by betting with Mr. Moore. But it benefits you with clear, understandable, and well-documented portable code that does not lock you into a particular environment – your MATLAB code can be cross-compiled onto any other MATLAB-supported processor.

Table 1 also shows that VSIP code offers a much smaller performance penalty. This is because Mercury tunes VSIP calls to the same level of detail as it invests on SAL. In fact, VSIP is a reasonable, standards-based substitute for SAL, in the areas supported by Mercury's commitment to VSIP Lite.

Identifying the performance deltas between standard functions and hand-optimized code is an important step in allocating programming investment. An understanding of application requirements and toolset performance lets developers apply the "80/20 rule" to coding. This guideline suggests that overcoming a small number of performance

bottlenecks yields a great amount of performance improvement, possibly as much as the economics of hand-coding justify both in terms of programming costs today and redeployment costs in the future. This is especially true in cases where an application requires many iterations of a function that suffers large performance penalties. Often, the code a developer creates for this function in one instance is transferable to other instances as well, providing a big return on a modest programming investment.

Developers must, however, carefully track and document instances of custom code created for a particular platform, especially where the code takes advantage of the platform's native features. This code is almost certain "to break" in future redeployments. Documentation of where these breaks are likely to happen, and what they may entail, helps managers plan redeployment projects and helps future developers identify the most likely things to fix.

5.0 Portability at Work

Existing standards can promote easy portability today. At the Exploitation Technology Symposium (ETS'98) exhibition in San Diego last year, Mercury demonstrated a RACE system comprised of 128, 200-MHz PowerPC 603e microprocessors, running three applications directly related to current military and intelligence programs. The demonstration included Multi-Scale Intelligent Bandwidth Compression (MS-IBC) and Real-Time, Space-Time Adaptive Processing (RT-STAP), both ported by engineers from the MITRE Corporation, and Image Formation Processing (IFP) ported by Mercury engineers. These ports were accomplished in one and one half days.

The software tools used to achieve these fast ports and scalability for the ETS'98 demo included a pre-release version of MPI for RACE developed by MPI Software Technology, Inc. (MSTI), Mercury's Parallel Application System (PAS™), and the Khoros operating system kernel. Additionally, Mercury's Scientific Algorithm Library (SAL) provided mathematical functions tuned by hand for optimal performance on the target RACE processors. These tools helped minimize development efforts and maximize performance throughput.

Mercury plans to introduce VSIP Lite functions for its Motorola PowerPC microprocessor-based products, including PowerPC 750 and future AltiVec™ technology systems. Calls to VSIP functions on one supported platform should run without modification on any other plaftorm that supports the standard. This will help drive down development and porting time and expenses.

6.0 Summary

Embedded applications often need all the performance they can achieve, through means such as hand-tuning individual functions to take advantage of features of target processors. This often results in a programming penalty by requiring the use of "non-standard" code.

Standard function calls, conversely, allow quicker programming and portability, but often imposes performance penalties because its provider may not implement it in a way that takes maximum advantage of a particular target platform.

Developers must assess the performance impact of standard function calls and compare it to the productivity impact of customizing all or parts of their applications. Then they must target which elements, if any, to tune by hand in order to meet performance requirements.

Developers should also influence hardware specifiers to select platforms from vendors with a strong commitment to supporting software standards. These vendors are the most likely to implement standards for maximum performance and provide forward code compatibility with future products. This will result in the best combination of programmer productivity and application redeployability on new generations of faster, smaller, less costly embedded processors.