On the BLAS and the BLAS Technical Forum

Andrew Lumsdaine
Laboratory for Computer Science and Engineering
University of Notre Dame

February 26, 1997

1 Introduction

As the work of the BLAS Technical Forum progresses, I have been concerned that (in my mind anyway) there is no clearly defined goal towards which we are all working. I am afraid that without some clearly defined goals that the work of the Forum will sink to irrelevance.

This document is my attempt to try to sort through the work of the past year and to propose some particular objectives for the forum.

Author's note: At this point, this document is sketchy. I have not yet completely fleshed out some of the points I would like to make here. I should have a more complete document by the time we convene in Dallas. Some of my other thoughts on BLAS matters have already appeared (and have been sent to the blast mailing list) as the BLAIS working note #0, "Standard Sequential Mathematical Libraries: Promises and Pitfalls, Opportunities and Challenges."

2 What is the BLAST Forum about?

So, what are we about? We all know how important the BLAS have been to scientific computing over the past years. When Jack convened the first BLAS Technical Workshop in November 1995 to standardize the BLAS (ala HPF or MPI), there was significant interest in this effort.

There seem to be several approaches that we can take. The first five that I list here form sort of a natural progression that seems to be the direction in which the forum is headed. I will argue against this approach and propose an alternative (Alternative #1).

Approach #0 (Default): Declare Level-1, Level-2, and Level-3 BLAS to be a standard. In many ways, the Level-1, Level-2, and Level-3 BLAS are a *de facto* standard for numerical linear algebra, anyway.

Approach #1: Augment Approach #0 with a few more functions and declare a standard.

Approach #2: Augment Approach #1 with the proposed sparse BLAS and declare a standard.

Approach #3: Augment Approach #2 with Scalapack (or Plapack) parallel BLAS and declare a standard.

Approach #4: Augment Approach #3 with some bindings for C and declare a standard.

Alternative #1: Take what we now know about all aspects of high performance computing (architecture, algorithms, applications, compilers, software engineering) and produce something completely new.

At this point, it seems that the BLAST forum is converging more or less on Approach #4. I would suggest that this is a suboptimal approach.

I would like to make a strong argument for Alternative #1 and suggest that we drop Approach #0-#4 and start from the ground up. In the next sections, I will discuss the state of high-performance computing and draw a profile of our expected users. From these discussions, I will draw some conclusions about what I think the BLAS forum should be doing and what the next generation BLAS should look like.

3 What is the state of the art?

"Supercomputing is dead." We've all heard that repeated countless times. "Supercomputing" has even been dropped from the name of the premier conference for high-performance computing — it is now simply named SC.

That supercomputing is dead reflects the fact that there is no longer a distinct market niche for that particular type of computing. High performance scientific computers are now large collections of commodity microprocessors. Witness ASCI.

We need to face the facts. The scientific computing community, and the associated hardware and software market, is relatively small. We are now the trickle-down beneficiaries of technologies developed for fields other than our own. There are profound ramifications to this truth, and we need to consider them.

Hardware and architecture. As mentioned above, high-performance computing is now based on, and will be based on, collections of commodity microprocessors. There are various flavors of how the processors are connected, distinguished primarily by how memory subsystems are shared (or not) among the processors. Other relevant features include RISC and multiple levels of cache.

Languages and compilers. Just as scientific computing is the user of trickle-down hardware technology, we will likely be the users of trickle-down software technology. That is, there is not a sufficient market in scientific computing to support hardware that is targeted only for scientific computing. Similarly, there is not a sufficient market in scientific computing to support software that is targeted only for scientific computing (and by this, of course, I mean any flavor of Fortran). Admittedly, the cost of supporting a compiler is not as large as that of supporting hardware, but the same principles apply. Hardware vendors (and third-party compiler developers) are going to go where the money is. That means C and C++. And Java, of course, but it remains to be seen whether Java is a serious language, much less a serious language for high-performance computing (I doubt the latter). Indeed, recent work (e.g., PhiPAC at Berkeley and my own Matrix Template Library) have shown that peak performance can be achieved with C and C++ in a portable way while providing powerful data abstractions.

Applications. Much of scientific computing is based on solving systems of nonlinear differential equations on irregular meshes. The resulting problems are characterized by sparsity and irregular structure.

4 Who are our customers?

Black-box users: There is a class of users who do not care about underlying data formats, algorithms, etc., but who want a rich set of high-performance subroutines for numerical linear algebra.

White-box users: There is a class of users who might want to roll their own BLAS, or tweak particular subroutines for the last few flops.

Implementors: Someone will need to actually implement BLAS. In a very real sense, these users are still our customers — the product of the forum is a standard, not an implementation.

5 Conclusions: What the new BLAS should be.

This is a set of conclusions I have drawn about what the new BLAS should be — and what the Forum should be concentrating on.

- The new BLAS should be defined in two layers: a user layer and a tiny layer. The tiny layer should consist of a small number of high performance computational kernels that can be composed to provide the high performance subroutines at the user level.
- The new BLAS should provide a rich set of well-defined language-neutral functions for performing numerical linear algebra. By well-defined, I mean that the function semantics and issues such as computational complexity, argument aliasing, and so forth should be explicitly defined.
- The new BLAS should provide language bindings that support the common programming idioms and styles of the target languages.
- The new BLAS should be unified. There should be no distinction between parallel, sequential, dense, or sparse.
- The design of the new BLAS should start top down. The comprehensive set of user-level functions should be defined first. This will delineate the data types, algorithms, and communication that need to be developed and supported by tiny BLAS.