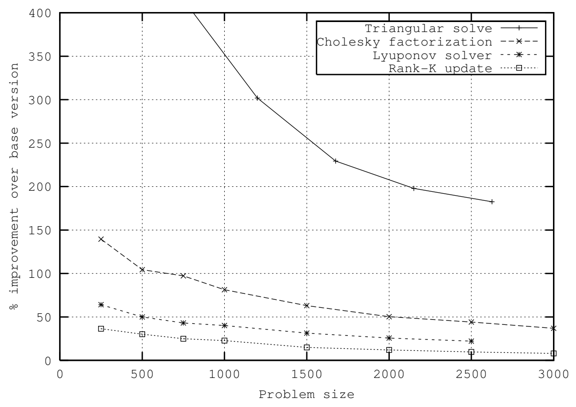
A Case Study in Exploiting Layers to Optimize Scientific Software

Samuel Z. Guyer Calvin Lin

Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712

This paper presents a case study in improving the performance of layered scientific software using *library-level optimization*. We augment our previous work to include a notion of layers, and apply the technique to the three layers that make up the PLAPACK parallel linear algebra library—a global application level, an internal layer, and an MPI message passing layer. We show how significant performance improvements of 10% to 180% for large matrices and 36% to 600% for small matrices are possible for four PLAPACK applications. Our approach works because it first exploits layer boundaries to facilitate high-level program analysis and optimization, and then systematically dissolves layer boundaries to expose new optimization opportunities. This work is presented in the context of the Broadway compiler system, which uses an annotation language to capture semantic information about the abstractions present in software libraries.



1 Introduction

Layering has long been used to simplify the design of software systems. Layering helps decompose systems into manageable pieces, and creates reusable modules [6]. Developers of large scientific software systems have come to depend more and more on layering, since such systems tend to represent a broad spectrum of specialized programming domains. For example, the POOMA framework consists of five layers [16], with the higher layers representing abstractions in the problem domain, such as solvers and complete simulations, and the lower layers representing abstractions in the implementation, such as communication, data distribution, and sequential kernels. Many other systems are similarly layered [4].

One well known problem with layers is the performance degradation that comes from not optimizing operations in an end-to-end manner. These issues have been addressed in certain domains, typically by leveraging domain-specific information about the layers. For example, systems have been introduced to optimize layered communication protocols [1] and toolkits for distributed computing [12]. There have been few attempts, however, to optimize layered scientific software, perhaps because such systems span such a broad range of domains. Instead, the thrust of such systems has been to optimize each layer or component independently.

This paper argues that *library-level optimization,* a compiler-based approach for optimizing library operations [10, 11], is a promising approach for optimizing lay-