An Hierarchical Approach for Performance Analysis of ScaLAPACK-Based Routines Using the Distributed Linear Algebra Machine

Krister Dackland and Bo Kågström

Department of Computing Science, University of Umeå, S-901 87 Umeå, Sweden.
Email addresses: dacke@cs.umu.se and bokg@cs.umu.se.

Abstract. Performance models are important in the design and analysis of linear algebra software for scalable high performance computer systems. They can be used for estimation of the overhead in a parallel algorithm and measuring the impact of machine characteristics and block sizes on the execution time. We present an hierarchical approach for design of performance models for parallel algorithms in linear algebra based on a parallel machine model and the hierarchical structure of the ScaLAPACK library. This suggests three levels of performance models corresponding to existing ScaLAPACK routines. As a proof of the concept a performance model of the high level $Q,R$ factorization routine PDGEQRF is presented. We also derive performance models of lower level ScaLAPACK building blocks such as PDGEQR2, PDLARFT, PDLARFB, PDLARFG, PDLARF, PDNRM2, and PDSCAL, which are used in the high level model for PDGEQRF. Predicted performance results are compared to measurements on an Intel Paragon XP/S system. The accuracy of the top level model is over 90% for measured matrix and block sizes and different process grid configurations.

1 Introduction

At PARA95 we presented an algorithm for reduction of a regular matrix pair $(A,B)$ to block Hessenberg-triangular form $(H,T)$ [3]. It was shown how to reorganize the elementwise algorithm to perform blocked factorizations and higher level BLAS operations. The objective was to develop an algorithm that showed good performance on computers with hierarchical memory and to make use of this effort to design an efficient parallel implementation. Presently, we are extending the blocked algorithm to a scalable and portable program by expressing most computations in terms of existing factorization and update routines from the ScaLAPACK library [1]. To facilitate the design and analysis of the algorithm we decided to develop a performance model.

Since the ScaLAPACK routines are intended to be black boxes for the user it would be appropriate to build a model on a library of performance models corresponding to existing and future ScaLAPACK routines. Indeed, we suggest that any library of routines for scalable high performance computer systems should also include a corresponding library of performance models. In this contribution we present an approach for design of performance models based on the
Distributed Linear Algebra Machine (DLAM) [1], and the hierarchical structure of the ScaLAPACK library. By use of lower level ScaLAPACK models it is straightforward to design new higher level models for new routines and applications based on ScaLAPACK.

The rest of the paper is organized as follows. Section 2 introduces the parallel machine model and the data layout for matrices and vectors. In Section 3 different level 1, 2 and 3 performance models are presented. Finally, in Section 4 we show predicted and measured timing results that verify the hierarchical approach for performance modeling.

2 Parallel Machine Model and Data Layout

Parallel Machine Model. DLAM is a theoretical model of a parallel computer dedicated to dense linear algebra. The machine characteristics of a target architecture are specified with a few parameters that define the costs for computations and communications.

A $P$-process DLAM consists of $P$ BLAS processes that communicate through a $P_r \times P_c$ BLACS network, which is a logical 2D grid with $P_r \cdot P_c \leq P$. Data are exchanged between BLAS processes by calling BLACS primitives. The processes can only perform BLAS [9, 7, 6] and BLACS [4, 5] operations.

In the DLAM model BLAS operations of the same level (1, 2 or 3) are assumed to have approximately the same performance measured in millions of floating point instructions per second (Mflop/s). So from a performance analysis perspective DLAM only distinguishes three different BLAS instructions and the execution times for level 1, 2, or 3 BLAS instructions are denoted $\gamma_1, \gamma_2,$ and $\gamma_3$, respectively.

BLACS support point-to-point communication, broadcast and combine operations along a row or column of the process grid. The time to transfer $n$ items between two processes is modeled as

$$T_s(n, \alpha, \beta) = \alpha + n\beta,$$

where $\alpha$ is the start up cost and $\beta$ is the per item cost (1/bandwidth). The corresponding models for broadcast and combine operations are

$$T_b(Top, p, n, \alpha, \beta) = K('\text{bcast}', Top, p, n)T_s(n, \alpha, \beta),$$

$$T_c(Top, p, n, \alpha, \beta) = K('\text{combine}', Top, p, n)T_s(n, \alpha, \beta),$$

where $p$ is the number of processes involved, $Top$ is the network topology to be emulated during communication and $K(\cdot)$ is a hardware dependent function. If the physical network supports a 2D mesh and collisions are ignored then $K('\text{bcast}', '1 - tree', p, n)$ is approximated as $\log_2(p)$. The parameter $Top = '1 - tree'$ means that there is only one branch at each step of the broadcast tree. The corresponding number for $K('\text{combine}', '1 - tree', p, n)$ is $2\log_2(p)$.