Chapter 4
Sparse Linear Algebra:
Applying New Features to Traditional Paradigms

ABSTRACT

This chapter shows several new programming strategies based on tasking to parallelize sparse linear algebra kernels. The reader will explore different approaches to improve the performance of these kernels thanks to a better workload distribution and comprehension of the data layout. This will be accomplished through the study of some of the most popular and widely used sparse operations, such as SpMV (sparse matrix vector multiplication), GTSV (triangular solve), or CG (conjugate gradient). Those strategies have been tested on multicore systems. Some of them equipped GPU devices, showcasing how to overcome the peculiarities of task-based parallelized kernels in the context of sparse linear algebra computations.

INTRODUCTION TO SPARSE LINEAR ALGEBRA

The input matrices used to solve specific engineering and scientific problems sometimes present a high number of zero values due to the nature of the problem to solve. In these situations, where most of their values are zeroes, the authors said the matrix is unbalanced and sparse. This characteristic of the matrices turns out in a non-uniform memory access pattern; thus those operations tend to be memory bound. Moreover, the sparsity of the data

DOI: 10.4018/978-1-7998-7082-1.ch004

makes it difficult to evenly distribute the workload among cores, which is the main target of the strategies presented in this chapter.

Specifically, different task-based approaches are presented in order to balance the workload and palliate the effect of the irregular memory access patterns. The authors focus on three widely-used operations in Sparse Linear Algebra: the sparse matrix vector product (SpMV), the general tridiagonal solve (GTSV) and the conjugate gradient (CG). Moreover, a multi-core CPU architecture, as well as a GPU, will be targeted.

SPARSE MATRIX-VECTOR MULTIPLICATION (SPMV)

The sparse matrix-vector product (SpMV) is defined as:

$$y := \alpha Ax + \beta y,$$

where x and y are dense vectors, α and β are scalars, and A is a sparse matrix. In this section, several strategies are proposed in order to parallelize the SpMV kernel, which operates on an input matrix stored in CSR format (Langr, 2016).

SpMV is characterized by being a highly unbalanced operation. Due to the sparse nature of the input matrix, the access pattern to the elements turns out to be non-uniform. This aspect has been tackled in literature from two main perspectives: proposing several storage formats for the input matrix and designing different implementations for the SpMV kernel in order to palliate this effect. This section is focused on the implementation of task-based strategies to parallelize the kernel and balance the computations among the cores. The reader will find, along with the explanation of the strategies, performance results for 12 matrices from the SuiteSparse Matrix Collection (Davis, 2011), formerly the University of Florida Sparse Matrix Collection, that illustrate the behavior of each technique. Matrices are named m1 to m12 and *cat*egorized as unbalanced (m4, m5, *m7* a*nd* m*12*) or *bal*anced matrices (m1-m3, m*6, m8*-m*11*), *depen*ding on the variability of non-zeros per row (See Figure 1). Moreover, reference results from Intel MKL single-threaded and multi-threaded SpMV kernels are included.