Ms. Rukhsana Shahnaz
(Ph.D. Scholar, DCIS, PIEAS)

Abstract

Fast solution of linear equations having the large sparse coefficient matrices is a vital requirement of advanced computations in science and engineering. The matrix-vector product (SMVP) is the most important kernel operation for theiterative linear solvers, and thus its performance has a key noted effect on the performance of linear solvers.

The efficient parallelization of SMVP is of prime significance to the scientific computing community. The main goal that is achieved with parallelization is to perform the matrixvector product faster than what is possible on a single processor. To achieve this on a distributed memory computer, we concentrate on minimizing the inter-processor communication, achieving a good balance of the workload, overlapping communication with computation along with optimizing single processor performance.

The thesis consists of two parts presenting the optimization and improvement of sparse matrix-vector multiplication performance on single as well as multi processors respectively. Operations on sparse matrices tend to exhibit lower performance on general purpose processors due to sparse nature of the data structure that generates large number of cache misses and large number of memory accesses. The performance degradation of sparse matrix algorithms is also due to the poor temporal and spatial locality. In a nut shell the performance of SMVP is highly dependent on the nonzero structure of the sparse matrix, the organization of the data and its computation. The research demonstrates that the reorganization of the matrix in terms of blocks and the resulting computation around the blocks of the matrix is valuable for optimization.

To alleviate the existent problems and to get performance improvement of SMVP on a single scalar processor, we propose two sparse storage formats, namely the grouped compressed row storage with permutation (GCRSP) and the blocked compressed row storage with permutation (BCRSP). The proposed formats are designed to efficiently exploit the benefits of blocking i.e. the block-based compressed row storage (BCRS), such as reduced indirect addressing, increased spatial and temporal locality along with eliminating the corresponding overheads (both zero-overhead of BCRS and loop-overhead of sparse block based compressed row storage, SBCRS). In an effort of achieving this goal, the approach used is to reorder the matrix entries before blocking to get maximum possible dense blocks.

The thesis reports an extensive performance study over variety of matrices taken from the matrix market. It has been observed that operating on the group of rows in GCRSP improves the performance over simple compressed row storage (CRS) and compressed row storage with permutation (CRSP) with an average of ***16%*** and ***25%***, respectively. Moreover, due to blocking in BCRSP, the performance improvements of an average of ***32%***, ***41%*** and ***20%*** are observed over CRS, CRSP and GCRSP respectively.

For the good load balancing and low communication cost, reordering of sparse matrices according to their sparsity structure is highly important. For this purpose we proposed reordering based partitioning strategies that tend to exploit sparsity of input matrix presenting the balanced load distribution along with the reduced communication cost. The proposed models permuted row (PR), permuted column (PC) and permuted matrix (PM) produce an average of ***49%*** better load balancing and ***14%*** better communication than the corresponding naïve row (NR), naïve column (NC) and checker board (CH) models. Moreover, they produce same level of balanced load and an average of ***78%*** better communication than the corresponding balanced naïve partitioning i.e. row block (RB), column block (CB) and balanced checker board (BCH) models. On the whole an average of ***30%*** performance gain for parallel SMVP is achieved by using BCRSP format along with permuted row (PR) partitioning over the implementation using CRS format with naïve row (NR) partitioning using cluster of eight processors.