Block Lanczos to Solve Integer Factorization Problem Using GPU’s

Harish Malla*, Vilas SantKaustubh, Rajasekharan Ganesh, and Padmavathy R.

National Institute of Technology
Warangal

Abstract. Public key cryptography is based on some mathematically hard problems, such as Integer Factorization and Discrete Logarithm problems. The RSA is based on Integer factorization problem. Number Field Sieve is one of the popular algorithms to solve these two problems. Block Lanczos algorithm is used in the linear algebra stage of Number Filed Sieve method for Integer Factorization. The algorithm solves the system of equations Bx=0 for finding null spaces in the matrix B. The major problems encountered in implementing Block Lanczos are storing the entire sieve matrix and solving the matrix efficiently in reduced time. Implementations of Block Lanczos algorithm have already been carried out using distributed systems. In the current study, the implementation of Block Lanczos Algorithm has been carried out on GPUs using CUDA C as programming language. The focus of the present work has been to design a model to make use of the high computing power of the GPUs. The input matrices are very large and highly sparse and so stored using coordinate format. The GPU on-chip memories have been used to reduce the computation time. The experimental results were obtained for the following problems; RSA100, RSA110, RSA120. From the results it can be concluded that a distributed model over GPUs can be used to reduce the iteration times for Block Lanczos.

Keywords: Public Key cryptography, RSA, Block Lanczos, GPUs.

1 Introduction

Public key cryptography is based on some mathematically hard problems. The popular RSA is based on integer factorization and the counterparts ElGamal and Diffie-Hellman are based on discrete logarithm problem. In number theory, integer factorization problem is to factor the given composite number into its factors. The problem is found to be hard when the factors are big primes. The best known method to solve the above problem is Number Field Sieve.

The Number Field Sieve consists of two steps, such as sieving and solving. The sieving phase generates a large and sparse matrix called as sieve matrix. The solving phase, first reduces the large size matrix into small and still sparse matrix and later solves the linear system of equations.

*Corresponding author.
The solving phase is the main bottle-neck in the overall process. In the literature, many algorithms are reported. The method proposed by Lanczos is widely known and attempted method, since it needs less memory and easily adoptable for large and sparse matrices.

Block Lanczos algorithm which is a modified version of Lanczos algorithm used in the linear algebra stage of Number Field Sieve (NFS) is proposed in [1]. This algorithm is one of the ideal candidates for parallelization. The algorithm uses subspaces instead of vectors for solving the sparse matrix generated in sieving stage, for finding null spaces. The subspaces are represented using matrices. The parallel implementation of Block Lanczos using Mondriaan partitioning for sparse matrices is discussed in [2]. In this he discussed about the global-local indexing mechanism, vector partitioning, sparse matrix partitioning, sparse matrix-vector multiplication, AXPY operations and dense vector inner product computation. Coppersmith et al., discussed how Block Lanczos is much competitive than Gaussian Elimination for solving linear system of equations [4]. The paper also discusses that the block operations performed in Block Lanczos reduces the 32 matrix-vector operations to one. Nathan Bell et. al., reported the different format of representation for sparse matrix to store and perform matrix operations on them efficiently [6]. The different formats given by the author are DIA, ELL, CSR, COO, hybrid format. The use of COO format shows very little variance in efficiency over different data and applications. They also discussed about how matrix operations can be performed efficiently on different matrix formats that have been discussed in their previous work [7].

In the present study Block Lanczos is implemented on GPUs. The GPUs have larger number of cores on a chip when compared to CPUs. Also the Arithmetic Logical Units (ALUs) in case of GPUs are much more than in CPUs. Many-coreprocessors, especially the GPUs, have high floating-point performance. As discussed in [4], Block Lanczos algorithm is one of the ideal candidates for parallelization. Also from [6] and [7] it can be inferred that the sparse matrix operations of Block Lanczos can be performed efficiently on GPUs using CUDA. These ideas provided the motivation for implementing the Block Lanczos algorithm on GPUs using CUDA C.

1.1 Integer Factorization

There are different methods for Integer Factorization like continued fraction method, quadratic sieve, and number field sieve. Integer factorization algorithms require several nonzero vectors \( x \) belonging to Galois field (\( \text{GF}(2)^n \)) such that a system of equations \( Bx = 0 \) is obtained, where \( B \) is a given \( m \times n \) matrix over the field \( \text{GF}(2) \). This matrix \( B \) is called sieve matrix and is usually very large and highly sparse with \( m < n \). Suppose, an integer \( M \) is to be factored, the quadratic sieve method finds congruence’s between \( a_j^2 \) and product of \( p_i \) raised to some exponents \( b_{ij} \), modulus \( M \). Here \( p_i \) are primes or -1 and the \( b_{ij} \) are exponents, which are mostly zeroes. The quadratic sieve method then tries to find \( S \subseteq \{ 1, 2, -\ldots, n \} \) such that both sides of the congruence.

\[
\prod_{j \in S} a_j^2 = \prod_{j \in S} \prod_{i=1}^m p_i^{b_{ij}} \pmod{M}
\] (1)