UDC 519.612

SOME ESTIMATIONS OF PERFORMANCE OF PARALLEL ALGORITHMS FOR SOLVING LARGE LINEAR SYSTEMS OVER GF(2)

© **Mikhail Alekseevich Cherepniov**
Moscow State University named after M.V. Lomonosov, Leninskiye gory, 1, Moscow, 119899, Russia, Candidate of Physics and Mathematics, Associate Professor Numbers Theory Department, e-mail: cherepniov@gmail.com

Key words: fast algorithms, sparce linear systems, parallel algorithms, computer algebra.
This topic explains how to estimate the running time and RAM volume required by programs of Wiedemann-Coppersmith algorithm, Montgomery's algorithm, some modifications of them and new algorithm when uploading multiple compute nodes and some other details of these algorithms.

1 Introduction

In principle, each operation of any algorithm can be considered for possible use of several similar sites to reduce time of their implementation. However, if the number of arithmetic operations is fixed, the most significant reduced time - this is where time falls proportional to the growing of compute nodes' number. It will be that, for example, if you search relations in the methods of the discrete logarithm or factorization problem based on the factor databases. However, for most complex algorithms top rated time of their work $T(N, n, d, c \ldots; s)$ depend on task and used equipment parameters: N, n, d, c and the number of used compute nodes s and may behave differently on different areas of argument changing. Often when you increase s above a certain threshold $s_0(N, n, d, c \ldots)$, depending on the task settings, time not only falls, but begins to grow again. That is because time for exchange between s computing nodes grows faster than of running time on each compute node separately reduce.

According to the author it is legitimately to raise the issue of calculating of values $s_0(N, d, n, c \ldots)$, and $T_0(N, d, n, c \ldots) = \min_s T(N, d, n, c \ldots; s) = T(N, d, n, c \ldots; s_0(N, d, n, c \ldots))$ for some model cluster. As a model it is logical today to take the cluster with an unlimited number of compute nodes for which runtime of one arithmetic operation with machine words, multiplied by the some constant c, equal to the runtime of one machine word's passing between nods.

In modern computer network runtime of passing between the processor and RAM memory have $c \approx 5$, and between the individual parts of RAM $c \approx 20$.

As known by author, option c is determined by the ratio between the frequencies of the processor and "bus which links compute nodes, portion of information bits transferred on the internal network communications and some other characteristics of the cluster. Let's also assume that on a cluster there exists a possibility of direct delivery between performing specific job computing nodes; and broadcast from fixed site to the sites of some group by binary tree, i.e. by the logarithm of the number of elements of this group transfers. In this article, all the time calculates in units equal to time per arithmetic operation with machine words in such model

cluster. We assume that time for conversion of matrix formats and delay time for preparing network is negligible.

The task will characterize by four parameters: $N \geqslant 2^{26}$ - size of the original matrix (maximum number of rows and columns), d - upper bound of the number of nonzero elements (units) in each row of this matrix, n - machine word length, c - parameter which was mentioned above. However, if the volume of carriage, in bits, is V, while sending time will compute by formula $\frac{c}{n}V$ units of time. One unit time is the time that takes one operation with machine words in our model cluster.

2 Notes about symmetric matrixes

Let $N, M \in \mathbb{N}, \mathbb{F} = GF(2)$ and we want to obtain solution of system of linear homogeneous equations

$$DX = 0, D \in \mathbb{F}^{M \times N}, X \in \mathbb{F}^{N \times n}, M < N, \tag{1}$$

where n – width of block, typically equals to machine word length (32 or 64).

At the beginning of Montgomery's algorithm, and new algorithm [1] select random block $Y \in \mathbb{F}^{N \times n}$ and consider linear system

$$AX = B, A \in \mathbb{F}^{N \times N}; B = AY, X \in \mathbb{F}^{N \times n}, \tag{2}$$

where $A = D^T D \in \mathbb{F}^{N \times N}$ is singular symmetric matrix. Note that any solution X_D of the system (1) allows you to build a X_A - solution of the system (2) by the formula

$$X_A = X_D + Y.$$

Back, if X_A - is a solution of system (2), than $D^T D(X_A - Y) = 0$. If $rangD = M$, it follows that $D(X_A - Y) = 0$. That is $X_A - Y$ - solution of the system (1).

Let $dim_1 X$ is a dimension of the intersection of space $\langle X \rangle$, formed by columns of block X and the kernel of a linear operator D. Let $dim_2 X$ is a dimension of the intersection of the space, formed by columns of block X and the kernel of a linear operator $A = D^T D$, and $dim'_2 X$ - dimension of the intersection of space $\langle X \rangle$ and the kernel of a linear operator $A' = DD^T$

Theorem 1 *For arbitrary* $X \in \mathbb{F}^{N \times n}$

1. $dim_1 X \geqslant dim_2 X - (M - rangD)$
2. $dim_1 D^T X \geqslant dim'_2 X - (M - rangD)$

Proof.

1. In accordance with the size of the matrix D we have $dimKerD^T = M - rangD$. By definition, the vector X is in $KerA$ if and only if it is in $KerD$ or $DX \in KerD^T \setminus \{0\}$. The maximum number of linearly independent vectors that meet the second of these conditions is not more than $dimKerD^T$. So corang $KerD$ in $KerA$ not more than $M - rangD$. Crossing with $\langle X \rangle$ we obtain inequality 1.

2. The number of linearly independent vectors among columns of block D^TX decrease compared to the number of linearly independent vectors among columns of block X no more than $dimKerD^T = M - rangD$. So corang $KerD^T$ in $KerA'$ no more than $M - rangD$. Crossing the $\langle X\rangle$ with $KerA'$, we get confirmation of 2.

The theorem is proved.

Note that according to the dimensions, matrix A in general have more linear independent vectors in the core than matrix A'. So we will prefer A.

Theorem 2 *Let Krylov space $\langle W\rangle$ built as a sum of $\langle W_i\rangle, i = 0, 1, \ldots, m$, with an initial unit type $W_0 = B = AY$, on the first $m-1$ of which A-scalar production is nonsingular. Let $\langle W_m\rangle$ - A-orthogonal to whole $\langle W\rangle$, in particular to itself. Let along with $AW_i, i = 0, 1, ..., m-1$, also calculated AW_m and X by the formula*

$$X = \sum_{i=0}^{m-1} W_i(W_i^TAW_i)^{-1}W_i^TB, \tag{3}$$

$dim\langle W_m\rangle = \mu > 0$. Then by not more than $(n+\mu-1)(n+\mu)N$ bitwise operations with probability at least $1-\frac{1}{2^n}$ (subject to statistical independence $\langle Y\rangle$ and $\langle W\rangle$) one can calculate the solution of system (2).

Proof.

Note, that by condition the space $\langle W_m\rangle$ A-orthogonal to the space $W = \langle W_0\rangle + \cdots + \langle W_m\rangle$, in particular A-orthogonal to itself. By the way, note that in practice dimension $\langle W_m\rangle$ is small (say 2). Also by the the condition $\langle W_m\rangle$ contains all vectors from Krylov space that are A-orthogonal to all this space. Then AW_m have the form $w_0 + w_1 + \cdots + w_m, w_i \subseteq \langle W_i\rangle, i = 0, 1, 2, \ldots, m$. When $j \in \{0, 1, \ldots, m-1\}$ we obtain a chain of equations

$$w_j{}^TAW_j = (w_0 + \cdots + w_m)^TAW_j = (AW_m)^TAW_j =$$

$$(W_m)^TA(AW_j) \subseteq (W_m)^TAW = \{0\}$$

that, due to non singularity of A-scalar product at W_j shows that $w_j = 0$ for $j = 0, 1, \ldots, m-1$, i.e. $AW_m \subseteq \langle W_m\rangle$.

If $\langle AW_m\rangle \neq \langle W_m\rangle$, the element from the kernel of the matrix A can be obtained by reduction of right part of equality $AW_m = Z$ to triangular form by $2\frac{(\mu-1)\mu}{2}N$ operations (The complexity of one elimination with columns of matrices $AW_m\|W_m, Z \in \mathbb{F}^{N\times(n+\mu)}$ estimated with value N).

Now let $\langle AW_m\rangle = \langle W_m\rangle$. From formula (3) we get by substitution

$$(W_j)^T(AX - B) = (W_j)^TB - (W_j)^TB = 0, j = 0, 1, ..., m-1,$$

$$(W_m)^T(AX - B) = (W_m)^TA(AX - B) = 0,$$

because $AX - B \subseteq W$. Therefore, in view of the A-invariance of Krylov space

$$(W_j)^TA(AX - B) = (AW_j)^T(AX - B) \subseteq$$
$$W^T(AX - B) = \{0\}, j = 0, 1, ..., m-1,$$

that is vector of the block $AX-B$ are A-orthogonal to W , i.e. $\langle AX-B\rangle \subseteq \langle W_m\rangle$. It means that linear operator A displays sum of spaces $\langle X-Y\rangle + \langle W_m\rangle$ in $\langle W_m\rangle$. If $\langle Y\rangle \not\subseteq \langle W\rangle$, dimension of the first space is larger than the dimension of the second, and with the assistance of the Gaussian exceptions in equality $A(X-Y\|W_m) = Z, \langle Z\rangle \subseteq \langle W_m\rangle$, item from the kernel of A can be found.

The probability that the $\langle Y\rangle \not\subseteq \langle W\rangle$ subject to the statistical independence of these spaces and singularity of matrix A , that leads to $\langle W\rangle \neq \mathbb{F}^N$, obviously estimated by value $1-\frac{1}{2^n}$, where n - the number of columns in Y . The number of eliminations with columns of matrices $X-Y\|W_m, Z \in \mathbb{F}^{N\times(n+\mu)}$ estimated with value $2((n+\mu-1)+...+1) = 2\frac{(n+\mu)(n+\mu-1)}{2}$. Theorem is proven.

Note that condition $W_0 = B$ in the last theorem may be replaced by $W_0 = Y$ without subject to statistical independence $\langle Y\rangle$ and $\langle W\rangle$. Then similarly we obtain $A(X-Y) \subseteq \langle W_m\rangle$, where by construction $\langle X-Y\rangle \subseteq \langle W\rangle \setminus \langle W_m\rangle$.

3 Parallelization of first and third phases

Before discussing the wording of theorems, refer that consistent implementations of considered algorithms picks up maximum time for repeatedly recurring operation of multiplying matrix and block of vectors, this is the main part of the first and third phases of algorithms. Therefore, parallelization must be primarily applied to this operation.

You can consider the two approaches to parallelization of multiplication matrix and block of vectors related to distributed storage of the matrix and (or) distributed storage of the block.

The need to use the first of these approaches is also the impossibility of storing in memory of one compute node all researched matrix D .

To use the second approach consider block of ns vectors. Namely, that all consider algorithms apply to a system of linear equations $DX = 0$, where $D \in \mathbb{F}^{M\times N}, X \in \mathbb{F}^{N\times ns}$. The amount s called block factor. Let for simplicity $M = N$.

Consider the task of parallel distributed multiplication of sparse matrix on block.

To ensure balance load its need to separate sparse matrix into parts, containing approximately the same number of units. Thus without additional transformation, matrix can be separated in one direction - by ratios. Divide matrix $D \in \mathbb{F}^{N\times N}$ of our system of linear homogeneous equations on horizontal strips by number l of used compute nodes. Thus processor with number i will receive matrix D_i , in which the nonzero left only N/l matrix rows of D .

In the first phase of Montgomery's and the new algorithm it is necessary to calculate $(D^TD)^iB$, $B^T(D^TD)^iB, B \in \mathbb{F}^{N\times ns}, i = 1, 2, \ldots$.

Have

$$D^TD = \sum_{i=1}^{l} D_i^TD_i,$$

$$D^TDB_j = \sum_{i=1}^{l} D_i^TD_iB_j, j = 1, \ldots, s, \tag{4}$$

where $B_j \in \mathbb{F}^{N\times n}$ - is j vertical strip of $B \in \mathbb{F}^{N\times ns}$. Storage of this strips and matrix D_i on the compute node need memory

$$Nn + \frac{nNd}{l}$$

bit (we assume that every non-zero matrix element D_i have the row number and column number in one machine word).

Thus distributed computation of block vector $D^T DB$ on ls compute nodes will require $\frac{N}{l}d$ operations for calculate items $D_i B_j$ and $\frac{Nd}{l}$ (number of nonzero elements in the matrix ${D_i}^T$) operations to complete the calculation $D_i^T D_i B_j \in \mathbb{F}^{N\times n}$.

Further, the algorithm consistently evaluated $(D^T D)^k B$. This requires sending with addition by formula (4) and inverse mailing received left part of this equality for all l using in its calculation computer nodes of computing site. Using the cyclic sending this takes time $\frac{c}{n}2Nn = 2Nc$.

To sum up, we get a time to calculate $(D^T D)B$ at ls compute nodes in the form

$$\frac{2Nd}{l} + 2Nc. \tag{5}$$

Calculation of the total Krylov space needed in the new algorithm and in algorithm of Wiedemann-Coppersmith, to build the solution. In addition to the first phase of the algorithms vectors that constituents Krylov space uses for constructing coefficients of series as scalar products $X^T D^i Y$ for algorithm of Wiedemann-Coppersmith, and $B^T (D^T D)^i B$ for the new algorithm.

For distributed by rows (strips) storage at l compute nodes total block B^T even need $\frac{Nns}{l}$ bits. After computing the $(D^T D)^i B_j$ on each of the l compute nodes of group with number $j, 1 \leqslant j \leqslant s$, it calculates its part of scalar production $B^T (D^T D)^i B_j$ by

$$3\frac{Nns}{l(log_2 \frac{N}{2} - log_2 log_2 \frac{N}{2})}$$

arithmetic operations (for corresponding algorithm one can see pp. 342-343 [4]). When $N \geqslant 2^{26}$ this fraction can be evaluated from the top by value

$$\frac{4Nns}{l \cdot log_2 N}. \tag{6}$$

All parts after concatenation inside one group gives $B^T (D^T D)^i B_j$, and for all groups - $B^T (D^T D)^i B$. Sending time is the minor due to small size of sent matrices. Adding to the evaluation (5) we obtain estimate of calculation time for the next block of vectors from Krylov space and the coefficient of the series on the cluster with sl computing nodes:

$$\frac{2dN}{l} + \frac{4Nns}{l \cdot log_2 N} + 2Nc. \tag{7}$$

To simplify the calculations we will use the value of l, that makes equal resulting components:

$$l = \frac{d}{c}. \tag{8}$$

A more detailed calculation shows that the second element in the estimation (7) when considered in the present values of parameters, namely, $2ns < dlog_2 N$ less than the third. For

large values of s it's better to do scalar multiplications on the additional $O(\frac{s}{log_2 N})$ compute nodes so that the second element of the evaluation (7) be less than the first and the third. So let's use estimation

$$4Nc \tag{9}$$

with overall estimation throughout the first phase of the $2\frac{N}{ns}$ steps approximately

$$\frac{8N^2c}{ns} \tag{10}$$

with condition $sl \leqslant C$ or $s\frac{d}{c} \leqslant C$ where C - general number of compute nodes of the cluster.

The memory of one node that participate in multiplication matrix on the block and calculating coefficients of the series, should have RAM space about

$$\underbrace{\frac{nNd}{l} + Nn + \frac{Nns}{l}}_{\text{matrix } D_i;\ \text{current } B_j\ ;\ \text{part of } B^T} = \frac{Nn}{8\cdot 10^9}(1 + c + \frac{sc}{d})\text{GB}. \tag{11}$$

when $l = \frac{d}{c}$. In the case of $N \approx 2^{26}$ this is approximately

$$\frac{1}{2}(1 + c + \frac{sc}{d})\text{GB}. \tag{12}$$

Note that if scalar multiplications calculates on another additional compute nodes, then memory on a single node enough demand up

$$\frac{1+c}{2}\text{GB}.$$

For obtaining general evaluation of the first and the third phase of the new algorithm in various applications, estimation (10) must be multiplied by a constant, not a great 2.

In the case of Widemann-Coppersmith algorithm rating for memory usage gives expression (12). Estimate for running time may be obtain similar to (7), where there is no factor 2 neer d, as there is no D^T. So l will be determined by the formula

$$l = \frac{d}{2c}.$$

For the time of the first phase, contains the calculation of the $\frac{2N}{ns}$ series coefficients and taking into account the need to build two passages (second to build the solution), we get very close to the previous estimation:

$$\frac{16N^2c}{sn}. \tag{13}$$

Note only that in the second pass of algorithm Widemann-Coppersmith, operation similar to multiplication by B^T replaced by right multiplying on a relatively small matrices g_i that requires equivalent time.

4 Parallelization of the second phase

In the algorithm of Widemann in version of Coppersmith vector $\bar{g}$ builds consistently increasing its size. Giorgi P., Jannerod C-P., Villard G. in [3] proposed another option with almost linear estimation of complexity, though with several logarithmic and rather big absolute multiplicative constants. So optimized algorithm we will call algorithm of Widemann-Coppersmith with matrix polynomial multiplication.

Let

$$h = h(\lambda) = \sum_{i=0}^{\infty} H_i \lambda^i, H_i \in \mathbb{F}^{ns \times ns}. \tag{14}$$

Here the size of ns selected so that the algorithm can be apply to the construction of approximation polynomial at the second stage of the algorithm Widemann-Coppersmith using block factor s.

Definition 1 *Degree of vector polynomial* $m(\lambda) \in (\mathbb{F}[\lambda])^{2ns \times 1}$ *called its degree as a polynomial with vector coefficients from* $\mathbb{F}^{2ns \times 1}$.

Definition 2 *Order of vector polynomial* $m(\lambda) \in (F[\lambda])^{2ns \times 1}$ *is a nonnegative integer* j, *that satisfies* $h(\lambda)m(\lambda) = \sum_{i \geqslant j+1} \mu_i \lambda^i, \mu_{j+1} \neq \overline{0}$.

Definition 3 σ*-basis of the series* $h(\lambda)$ *let's call matrix polynomial* $M(\lambda) \in \mathbb{F}^{2ns \times 2ns}[\lambda]$ *that satisfies the following conditions:*

1. The columns $M^{(i)}(\lambda)$ *of matrix* $M(\lambda)$ *have an order of not less than* σ.

2. Any $v \in \mathbb{F}^{2ns \times 1}[\lambda]$, *whose order is not less than* σ, *have unique representation*

$$v = \sum_{i=1}^{2ns} M^{(i)} C^{(i)}, C^{(i)} \in \mathbb{F}[\lambda], degM^{(i)} + degC^{(i)} \leqslant degv.$$

To implement Widemann-Coppersmith algorithm, we must build Pade approximations to series of the form (14), where $H_i = X^T D^i Y; X, Y \in \mathbb{F}^{N \times ns}$, namely $P(\lambda), Q(\lambda) \in \mathbb{F}^{ns \times 2ns}[\lambda]$, that satisfies

$$(h \parallel I_{ns}) \begin{pmatrix} Q \\ P \end{pmatrix} = O(\lambda^{2d+1}), degQ, degP \leqslant d, \tag{15}$$

where I_{ns} in concatenation is identity matrix from $\mathbb{F}^{ns \times ns}, d = \frac{N}{ns}$.

Let further for simplicity $Nns = 2^k$.

Theorem 3 *Let* $N \geqslant 2^{26}$. *An upper bound for running time of parallel implementation using block factor* s *of the second phase of the Wiedemann-Coppersmith algorithm with matrix polynomial multiplication*

1) When

$$s \geqslant \frac{120(log_2 log_2 32N)(log_2 \frac{ns}{2} - log_2 log_2 \frac{ns}{2})}{17n} \tag{16}$$

has the form :

$$1122Nn^2s^3(log_2 32N)(log_2 4N). \tag{17}$$

2) When

$$s \leqslant \frac{120(log_2 log_2 32N)(log_2 \frac{ns}{2} - log_2 log_2 \frac{ns}{2})}{17n} \tag{18}$$

has the form :

$$15840Nns^2(log_2 32N)(log_2 4N)(log_2 log_2 32N). \tag{19}$$

Remark 1 *Because inequality* $z \geqslant 2Alog_2 A$ *leads to* $z \geqslant Alog_2 z$ *, we obtain that if* $s \geqslant 2\frac{120log_2 log_2 32N}{17n} log_2\left(\frac{120log_2 log_2 32N}{34}\right)$ *, than inequality (16) holds.*

Proof of 3.

To build the necessary approximation we use the algorithm from article [3], which builds σ-bases, consistently doubling σ. According to this article we can built whole basis, when replace series $h(\lambda)$ to $h(\lambda^{2ns}) \times (1, \lambda, \lambda^2, \ldots, \lambda^{2ns-1})^T$ and construct approximation to it which order is $2dns$, using $2d$ coefficients of series $h(\lambda)$. Estimation of running time of program of corresponding algorithm $C(ns, ns, 2dns)$ can be obtained from the proof of theorem 2.4 [3], replacing d to $2dns$, namely

$$C(ns, ns, 2dns) \leqslant 2C(ns, ns, dns) +$$
$$MM(ns, dns) + MM(ns, 2dns),$$

where $MM(a, b)$-complexity of multiplying the two matrix polynomials with degree b from $\mathbb{F}^{a\times a}[\lambda]$. Continuing similarly, obtain

$$\begin{gathered} C(ns, ns, 2dns) \leqslant \\ MM(ns, 2dns) + \sum_{i=1}^{log_2 2dns} MM(ns, 2^{-i}2dns)(2^i + 2^{i-1}) \\ \leqslant \frac{3}{2}\sum_{i=0}^{log_2 2dns} 2^i MM(ns, 2^{-i}2dns). \end{gathered} \tag{20}$$

Remember, that for simplicity we demand $Nns = 2^k$.

Optimized algorithm to multiply matrix polynomials may be taken from the article [5]. According to lemma 3.2 of this work obtain following estimation

$$MM(ns, 2^i) \leqslant \alpha_Q \frac{(ns)^2}{n} + \beta_Q \frac{(ns)^2 \frac{ns}{n}}{log_2 \frac{ns}{2} - log_2 log_2 \frac{ns}{2}},$$

where $Q : \varphi(r^Q) \geqslant 2(2^i + 1), H : 2^H + 2 \leqslant Q \leqslant 2^{H+1} + 1$, $\alpha_Q \leqslant r^Q 2^H((6r + 2)\mu_r(H + 1) + \frac{2\alpha_2}{r^2}), \beta_Q \leqslant r^Q 2^H \beta_2$, where as r any natural number other than 2 can be selected, μ_r-sum of modules of the coefficients in a cyclotomic polynomial with number r, α_2 and β_2 respectively is the number of additions and multiplications for multiplication polynomials with degree $\varphi(r^2) - 1$. It is known that when $r = 3$ one can choose β_2 equals to 17. Here we apply trivial algorithm for addition of matrixes, broken by rows in machine words with the length of n and optimized algorithm of multiplying such matrixes as set out above. When $r = 3$ we get $\mu_r = 3$. Q can be chosen the lowest such that $3^{Q-1} \geqslant 2^i + 1$ (see. p. 8 [3]), so $3^Q \leqslant 9(2^i + 1)$, and when $i \geqslant 1$ we obtain $3^Q \leqslant 2^{i+4}$, and because $3log_3 2 < 2$, we have:

$$MM(ns, 2^i) \leqslant \frac{(ns)^2}{n} 2^{i+4} log_3 2^{i+4} \cdot$$
$$\left(20 \cdot 3(log_2 log_3 2^{i+4} + 1) + \frac{2}{9}\alpha_2 + \frac{17ns}{log_2 \frac{ns}{2} - log_2 log_2 \frac{ns}{2}}\right)$$
$$\leqslant \frac{(ns)^2}{n} \frac{2^{i+5}}{3}(i+4) \cdot$$
$$\left(60(log_2 \frac{2(i+4)}{3} + 1) + \frac{2}{9}\alpha_2 + \frac{17ns}{log_2 \frac{ns}{2} - log_2 log_2 \frac{ns}{2}}\right)$$

Because inequality

$$\frac{2}{9}\alpha_2 + 60 log_2 \frac{4(i+4)}{3} \leqslant 120 log_2 log_2 32dns$$

for $i \leqslant log_2 2dns$ obviously done, subject estimation can be continued by value

$$ns^2 2dns 11(log_2 2dns + 4) \cdot \left(120 log_2 log_2 32dns + \frac{17ns}{log_2 \frac{ns}{2} - log_2 log_2 \frac{ns}{2}}\right) \tag{21}$$

Thus

$$C(ns, ns, 2dns) \leqslant$$
$$\frac{3}{2} ns^2 22 \cdot 2dns(log_2 2dns + 4) log_2 4dns \cdot \left(120 log_2 log_2 32dns + \frac{17ns}{log_2 \frac{ns}{2} - log_2 log_2 \frac{ns}{2}}\right) \leqslant$$
$$66Nns^2 log_2 32N log_2 4N \cdot \left(120 log_2 log_2 32N + \frac{17ns}{log_2 \frac{ns}{2} - log_2 log_2 \frac{ns}{2}}\right) \tag{22}$$

In the case 1) this inequality can be continued by value

$$\frac{132Nns^2(log_2 32N)(log_2 4N) \cdot 17ns}{log_2 \frac{ns}{2} - log_2 log_2 \frac{ns}{2}}.$$

Since in this case, $\frac{ns}{2} \geqslant 16$, this evaluation can be continued by value

$$1122Nns^2(log_2 32N)(log_2 4N).$$

Second case obviously follows from the evaluation (22). Theorem proven.

Theorem 4 *Running time estimation of parallel implementations using block factor of Wiedemann-Coppersmith algorithm with the matrix polynomial multiplication under condition that* $N \geqslant 2^{26}$ *and*

$$N > \frac{71(log_2 32N)(log_2 4N)}{cn} \cdot \left(\frac{240 log_2 log_2 32N log_2(\frac{60}{17n} log_2 log_2 32N)}{17}\right)^4$$

has the form

$$94N^{1+\frac{3}{4}}n^{-\frac{1}{4}}c^{\frac{3}{4}}((log_2 32N)(log_2 4N))^{\frac{1}{4}}.$$

Proof.
Proof of this theorem follows from remark 1, theorem 3, 1) and evaluation (13) when

$$s = \frac{1}{4}\left(\frac{8Nc}{561n^3(log_2 32N)(log_2 4N)}\right)^{\frac{1}{4}}.$$

Note that when $N \geqslant 2^{26}, c \geqslant 1$ estimation of this theorem holds.

5 Conclusion

It's worth noting that in procedures (see [6]) using which 12 December 2009 was received new record of integer factorization, actually applies the usual algorithm of Widemann in Coppersmith version with $s = 8$, but construction of approximations was not consistent, but by binary tree as described above. Complexity of the corresponding algorithm proposed by Thom é in the [7] estimates by value

$$O(n^2 s^2(ns + log_2 k)k \cdot log_2 k \cdot log_2 log_2 k), k = \frac{N}{ns},$$

that is,

$$O(Nns(ns + log_2 N)log_2 N log_2 log_2 N).$$

This estimate is very close to evaluation (22). In principle, the difference is that degree of s is one less. Therefore, similar reasoning, when $s = O(N^{\frac{1}{3}})$ we can obtain time estimation of the type

$$O(N^{1+\frac{2}{3}}(log_2 N log_2 log_2 N)^{\frac{1}{3}}).$$

It is important to note that the necessary memory volume in this case

$$O(k(ns)^2 log_2 k) = O(Nnslog_2 \frac{2N}{ns})$$

bits, where multiplier $log_2 k$ associated with increasing integer factors when the recursive application of Fourier transform for polynomials with integer coefficients (see, for example [8, 9]) is used. For values of parameters for which was done record calculations $N \approx 3 \cdot 2^{26}, s = 8$, it's around 1TB. With using optimal value of s this is $O(N^{1+\frac{1}{3}} n log_2 N)$ that is really significantly much. An important advantage of algorithm [1] is the lack of need for multiplication of matrix polynomials to calculate only few coefficients such works. That leads to the good parallelization and runtime approximation for the second phase about $O(N)$ with the same asymptotic for RAM volume (with constant in O equals approximately 3).

However, some optimizations of this algorithm allows quite remove requirement growing RAM, and get an estimation of time of parallel implementations using block factor when $N \geqslant 2^{26}, n = 64, 1 \leqslant c$ of the whole algorithm of the type:

$$8,4c^{\frac{2}{3}}N^{1+\frac{2}{3}} \tag{23}$$

in assessing the number of compute nodes asymptotically $O(\frac{N^{\frac{2}{3}}}{logN})$ (when $N \approx 2^{26}, c \approx 12, d \approx 2^9$ number of required compute nodes is approximately 2400).

It is known that in usual parallel implementation of Montgomery's algorithm [2] the main complexity gives sending time during multiplying a matrix by a vector (see estimation (5)). You can assess that value by

$$\frac{N}{n}(\frac{c}{n}2Nn) = \frac{2cN^2}{n}.$$

As is known for the author, now there exist parallel implementations of Montgomery's algorithm with better bounds. Attitude this time work to the optimal time for the new algorithm roughly equal to 3. Increasing the size of the matrix N the attitude will be equal to

$$3\sqrt[3]{\frac{N}{2^{26}}}.$$

In addition, this algorithm have not "high parallelism", i.e. its parts cannot be done on independent clusters. The new algorithm and algorithm of Wiedemann-Coppersmith allow you to compute the coefficients of series and its approximations on the independent sites.

Note ones more feature of Widemann-Coppersmith algorithm and the new algorithm. When you use block factor s, running time of the first and third phases can be reduced to s, running time of the second phase increases depending on s not slower than s^2. For example, the complexity of multiplying two matrices from $GF(2)$ with size $ns \times ns$, as we saw above, have the estimate n^2s^3. At the same time, if this matrix divide to the blocks with the size $ns^{\frac{2}{3}} \times ns^{\frac{2}{3}}$ and send each pair of such blocks from different matrices for multiplication on an execution site, while time on sending and arithmetic operations will be about $O(n^2s^2)$, and the number of used compute nodes $O(s^{\frac{4}{3}})$. Similar considerations can be done fore the parallelization of scalar production of vectors. Solving of homogeneous systems (bring to the triangular form) can be made using a recursive algorithm that transfers calculations again to a matrix multiplication. Such parallelization leads to overall estimate of the form

$$O(\frac{N^2}{s}) + O(Ns),$$

that for $s = O(\sqrt{N})$ leads to $O(N^{\frac{3}{2}})$. The total number of compute nodes $2sl$ (see. (8)) will be about $O(N/logN)$.

We gather results in the table:

	Running time	**Nodes**	**Total RAM**
Wiedemann-Coppersmith with matrix polynomial multiplication	$94N^{1+\frac{3}{4}}n^{-\frac{1}{4}}c^{\frac{3}{4}}((log_2 32N)(log_2 4N))^{\frac{1}{4}}$	$O\left(\frac{N^{\frac{1}{2}}}{(log_2 N)^2}\right)$	$O(N^{1+\frac{1}{4}})$
Wiedemann-Coppersmith-Thomé	$O(N^{1+\frac{2}{3}}(log_2 N log_2 log_2 N)^{\frac{1}{3}})$	$O(\frac{N^{\frac{2}{3}}}{log_2 N})$	$O(N^{1+\frac{1}{3}}log_2 N)$
Montgomery (1995)	$\frac{2c}{n}N^2$	$O(1)$	$O(N)$
New algorithm	$8,4c^{\frac{2}{3}}N^{1+\frac{2}{3}}$	$O(\frac{N^{\frac{2}{3}}}{log_2 N})$	$O(N^{1+\frac{1}{3}})$

References

1. *Cherepnev M.A.* Block Lanczos-type algorithm for solving sparse linear systems //Diskr. Math. (in Russian). 2008. V. 20. N. 1. P. 145-150.

2. *Montgomery P.L.* A Block Lanczos Algorithm for Finding Dependencies over GF(2)//EUROCRYPT'95, LNCS, 1995. V. 921. P. 106-120.

3. *Giorgi P., Jannerod C-P., Villard G.* On Complexity of Polynomial Matrix Computations//ISSAC'03, August 3-6, 2003. Philadelphia, USA.

4. *Coppersmith D.* Solving Homogeneous Linear Equations Over GF(2) via Block Wiedemann Algorithm//Mathematics of Computation. Jan. 1994. V. 62. N. 205. P. 333-350.

5. *Cantor D., Kaltofen E.* On Fast Multiplication of Polynomials Over Arbitrary Algebras//Acta Informatica.1991. V. 28. P. 693-701.

6. *Kleinjung T., Lenstra A.K., and others* Factorization of a 768-bit RSA modulus. version 1.0. January 7, 2010. URL: http://eprint.iacr.org/2010/006.pdf.

7. *Thomé E.* Subquadratic computation of vector generating polynomials and improvement of the block Wiedemann algorithm//J. of Symbolic Computation. 2002. V. 33. Issue 5. P. 757-775.

8. *Vasilenko O.N.* Number-Theoretic Algorithms in Cryptography //AMS. Moscow State University, 2007.

9. *Naudin P., Quitte C.* Algorritmique Algebrique. Masson, 1992.

Accepted for publication 7.06.2010.

НЕКОТОРЫЕ ОЦЕНКИ ПРОИЗВОДИТЕЛЬНОСТИ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ РЕШЕНИЯ БОЛЬШИХ ЛИНЕЙНЫХ СИСТЕМ НАД $GF(2)$

© **Михаил Алексеевич Черепнев**
Московский государственный университет им. М.В. Ломоносова, Ленинские горы, 1, Москва, 119899, Россия, кандидат физико-математических наук, доцент кафедры теории чисел, e-mail: cherepniov@gmail.com

Ключевые слова: разреженные системы; факторизация целых чисел; параллельные алгоритмы; компьютерная алгебра.

В данной работе даны оценки времени, памяти при оптимальном числе узлов для известных блочных алгоритмов решения разреженных систем над $GF(2)$, их модификаций и нового алгоритма.