ISSN (Print) 2616-7263
ISSN (Online) 2663-1261

Л.Н. Гумилев атындағы Еуразия ұлттық университетінің

ХАБАРШЫСЫ

BULLETIN
of L.N. Gumilyov Eurasian
National University

ВЕСТНИК
Евразийского национального
университета имени Л.Н. Гумилева

ТЕХНИКАЛЫҚ ҒЫЛЫМДАР ЖӘНЕ ТЕХНОЛОГИЯЛАР сериясы

TECHNICAL SCIENCES AND TECHNOLOGY Series

Серия **ТЕХНИЧЕСКИЕ НАУКИ И ТЕХНОЛОГИИ**

№2(127)/2019

1995 жылдан бастап шығады
Founded in 1995
Издается с 1995 года

Жылына 4 рет шығады
Published 4 times a year
Выходит 4 раза в год

Нұр-Сұлтан,2019
Nur-Sultan,2019
Нур-Султан,2019

Редакцияның мекенжайы: 010008, Қазақстан, Нұр-Сұлтан қ., Сәтбаев к-сі, 2, Л.Н. Гумилев атындағы Еуразия ұлттық университеті, 349 б.
Тел: +7 (7172) 709-500 (ішкі 31-428). E-mail: *vest_techsci@enu.kz*

Жауапты хатшы, компьютерде беттеген: А. Нұрболат

Л.Н. Гумилев атындағы Еуразия ұлттық университетінің Хабаршысы.
ТЕХНИКАЛЫҚ ҒЫЛЫМДАР ЖӘНЕ ТЕХНОЛОГИЯЛАР сериясы
Меншіктенуші: ҚР БжҒМ "Л.Н. Гумилев атындағы Еуразия ұлттық университеті" ШЖҚ РМК
Мерзімділігі: жылына 4 рет
Қазақстан Республикасының Ақпарат және коммуникациялар министрлігінде 27.03.2018ж.
№16991 -ж тіркеу куәлігімен тіркелген
Тиражы: 25 дана
Типографияның мекенжайы: 010008, Қазақстан, Нұр-Сұлтан қ., Қажымұқан к-сі 12/1
Л.Н. Гумилев атындағы Еуразия ұлттық университеті
Тел: +7 (7172)709-500 (ішкі 31-428). Сайт: *http://bultech.enu.kz*

Editorial address:
2, Satpayev str., of. 349, L.N. Gumilyov Eurasian National University, Nur-Sultan, Kazakhstan, 010008
Tel.: +7 (7172) 709-500 (ext. 31-428), E-mail: vest_techsci@enu.kz

Responsible secretary, computer layout: Aizhan Nurbolat

Bulletin of L.N. Gumilyov Eurasian National University.
TECHNICAL SCIENCS and TECHNOLOGY Series
Owner: Republican State Enterprise in the capacity of economic conduct "L.N. Gumilyov Eurasian National University" Ministry of Education and Science of the Republic of Kazakhstan
Periodicity: 4 times a year
Registered by the Ministry of Information and Communication of the Republic of Kazakhstan
Registration certificate №16991-ж from 27.03.2018. Circulation: 25 copies
Address of Printing Office: 12/1 Kazhimukan str., L.N. Gumilyov Eurasian National University, Nur-Sultan, Kazakhstan 010008
Tel: +7 (7172) 709-500 (ext.31-428). Website: http://bultech.enu.kz

Адрес редакции: 010008, Казахстан, г. Нур-Султан, ул. Сатпаева, 2, Евразийский национальный университет имени Л.Н. Гумилева, каб. 349
Тел: +7(7172) 709-500 (вн. 31-428). E-mail: *vest_ techsci@enu.kz*

Ответственный секретарь, компьютерная верстка: А. Нурболат

Вестник Евразийского национального университета имени Л.Н. Гумилева. Серия ТЕХНИЧЕСКИЕ НАУКИ И ТЕХНОЛОГИИ
Собственник: РГП на ПХВ "Евразийский национальный университет имени Л.Н. Гумилева" МОН РК
Периодичность: 4 раза в год
Зарегистрирован Министерством информации и коммуникаций Республики Казахстан
Регистрационное свидетельство №16991-ж от 27.03.2018 г.
Тираж: 25 экземпляров. Адрес типографии: 010008, Казахстан, г. Нур-Султан, ул. Кажимукана, 12/1,
Евразийский национальный университет имени Л.Н. Гумилева
Тел.: +7(7172)709-500 (вн.31-428). Сайт: *http://bultech.enu.kz*

Л.Н. ГУМИЛЕВ АТЫНДАҒЫ ЕУРАЗИЯ ҰЛТТЫҚ УНИВЕРСИТЕТІНІҢ ХАБАРШЫСЫ. ТЕХНИКАЛЫҚ ҒЫЛЫМДАР ЖӘНЕ ТЕХНОЛОГИЯЛАР сериясы
№2(127)/2019

МАЗМҰНЫ

BULLETIN OF L.N. GUMILYOV EURASIAN NATIONAL UNIVERSITY.
TECHNICAL SCIENCE AND TECHNOLOGY SERIES
№2(127)/2019

CONTENTS

ВЕСТНИК ЕВРАЗИЙСКОГО НАЦИОНАЛЬНОГО УНИВЕРСИТЕТА ИМЕНИ Л.Н.ГУМИЛЕВА. СЕРИЯ ТЕХНИЧЕСКИЕ НАУКИ И ТЕХНОЛОГИИ

№2(127)/2019

СОДЕРЖАНИЕ

МРНТИ 27.25.19

N.M. Kassymbek [1], M.B. Mustafin [1], T.S. Imankulov [1], D.Zh. Akhmed-Zaki [2]

[1] *Al-Farabi Kazakh National University, Nur-Sultan, Kazakhstan*
[2] *University of International Business, Nur-Sultan, Kazakhstan*
(E-mail: nurislam.kassymbek@gmail. com, astanaforeverever@gmail.com, imankulov.timur@gmail.com, darhan.ahmed-zaki@kaznu.kz)

Optimization of the program for solving oil displacement problem

Abstract: This article discusses the optimization methods of the program to solve the problem of oil displacement. We consider the acceleration of program execution time on a single processor without the use of parallel architectures. The two-dimensional Buckley-Leverett model, which was solved by the Jacobi method, was chosen as the oil displacement problem. Program optimization is performed by three levels, the main of which is vectorization. After each level of optimization, an analysis of the program execution time is made and compared with a sequential program. All three levels of optimization showed an acceleration of about 89-95%.

Keywords: oil displacement, optimization, optimization keys, vectorization, SIMD extensions, SSE Intrinsics.

DOI: https://doi.org/10.32523/2616-7263-2019-127-2-40-48

Introduction. Numerical models of modern problems of industry and science work with very large data. It takes a lot of processor time to compute. To reduce computation time, parallel computing technologies like MPI, OpenMP, CUDA, etc. are used. But for parallel computing, we need a system with several computational processors. But it makes sense to optimize the program for one processor. This work is devoted to optimizing the program, using the potential of modern processors. When optimizing programs, the compiler can play a significant role, the better the compiler copes with the optimization, the faster your program can run. Accelerating the running time of programs can be achieved using vector extensions of processors, such as SSE, AVX and other versions. And there are also other optimization methods that can be implemented based on the work and purpose of the program. Since execution time is one of the important values, optimization plays a significant role in the development of programs.

Purpose of the work. The task in this paper was to first develop an algorithm for solving the problem of displacement of oil and write a program of this algorithm. Then optimize the program by runtime so that the program works as quickly as possible.

Methods

0.1. **Methods of solution.** To solve the problem of oil displacement, the Buckley-Leverett model was chosen [1]. Consider the case when a certain amount of water passes through the injection well within a certain time. In the injection and production wells, the pressures P_{inj} and P_{prod} ($P_{inj} > P_{prod}$) are given. Water flowing through the injection well displaces oil in the formation, which in turn moves toward the production well. It is necessary to develop a computer model of mass-exchange processes in the reservoir. Below is the layout of the reservoir:

The mathematical model of two-phase filtration consists of the equation of the balance of water and oil. Through the boundaries $\partial\Omega$ for the domain Ω the equation of the system will be in the following form [2]:

$$m\frac{\partial s_1}{\partial t} + div\,(\overrightarrow{v}_1) = 0$$

$$m\frac{\partial s_2}{\partial t} + div\,(\overrightarrow{v}_2) = 0$$

$$\overrightarrow{v}_1 = -k\frac{f_1}{\mu_1}\nabla P$$

$$\overrightarrow{v}_2 = -k\frac{f_2}{\mu_2}\nabla P$$

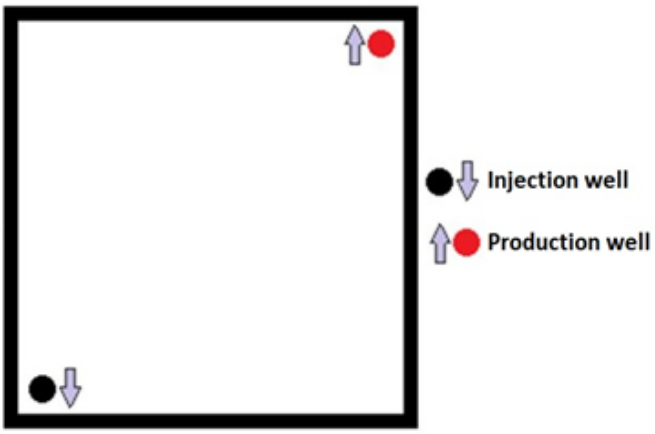

Figure 1 – Well Scheme

Here, m is the porosity of the reservoir, s_1, s_2 is the saturation of water and oil, $s_1 + s_2 = 1$, $\overrightarrow{v}_1$, $\overrightarrow{v}_2$ – velocities, k is the absolute permeability, f_1 and f_2 is the relative phase permeabilities, μ_1, μ_2 – viscosity, P – pressure. Thus, it is necessary to find the function P, the pressure, using the following initial conditions

$$P|_{t=0} = P_{rsv}$$

For the boundary conditions, the symmetry condition was taken:

$$\left.\frac{\partial P}{\partial x}\right|_{x=0} = \left.\frac{\partial P}{\partial x}\right|_{x=1} = 0$$
$$\left.\frac{\partial P}{\partial y}\right|_{y=0} = \left.\frac{\partial P}{\partial y}\right|_{y=1} = 0$$

Differential form of the equation:

$$m\frac{\partial s_1}{\partial t} + \frac{\partial}{\partial x}\left(-k_x\frac{f_1}{\mu_1}\frac{\partial P}{\partial x}\right) + \frac{\partial}{\partial y}\left(-k_y\frac{f_1}{\mu_1}\frac{\partial P}{\partial y}\right) = 0$$
$$m\frac{\partial s_2}{\partial t} + \frac{\partial}{\partial x}\left(-k_x\frac{f_2}{\mu_2}\frac{\partial P}{\partial x}\right) + \frac{\partial}{\partial y}\left(-k_y\frac{f_2}{\mu_2}\frac{\partial P}{\partial y}\right) = 0$$
$$\frac{\partial}{\partial x}\left(M_x\frac{\partial P}{\partial x}\right) + \frac{\partial}{\partial y}\left(M_y\frac{\partial P}{\partial y}\right) = 0 \tag{1}$$

First all equations are reduced to a dimensionless form. We give dimensionless variables to the following relations:

$$\bar{x} = \frac{x}{L_x}\,;\, \bar{k} = \frac{k}{k_c}\,;\, \bar{\mu_1} = \frac{\mu_1}{\mu_c}\,;\, \bar{\mu_2} = \frac{\mu_2}{\mu_c}\,;\, \bar{P} = \frac{P}{P_c}\,;\, \bar{t} = \frac{t}{T}\,;$$

For computer calculations, the oil displacement equation (1) was reduced to a finite-difference form and using the iterative method obtained the following formula [3]:

$$(M_x)_{i+\frac{1}{2}j}\frac{P^n_{i+1j} - P^{n+1}_{ij}}{h_1^2} - (M_x)_{i-\frac{1}{2}j}\frac{P^{n+1}_{ij} - P^n_{i-1j}}{h_1^2} +$$
$$(M_y)_{ij+\frac{1}{2}}\frac{P^n_{ij+1} - P^{n+1}_{ij}}{h_2^2} - (M_y)_{ij-\frac{1}{2}}\frac{P^{n+1}_{ij} - P^n_{ij-1}}{h_2^2} = 0 \tag{2}$$

From equation (2) it is possible to obtain the following formula for calculating the pressure at each point of the reservoir:

$$P^{n+1}_{ij} = \frac{(M_x)_{i+\frac{1}{2}j}P^n_{i+1j} + (M_x)_{i-\frac{1}{2}j}P^n_{i-1j} + (M_y)_{ij+\frac{1}{2}}P^n_{ij+1} + (M_y)_{ij-\frac{1}{2}}P^n_{ij-1}}{(M_x)_{i+\frac{1}{2}j} + (M_x)_{i-\frac{1}{2}j} + (M_y)_{ij+\frac{1}{2}} + (M_y)_{ij-\frac{1}{2}}} \tag{3}$$

Using the data in the n-th time layer, we find the pressures p_{ij}^{n+1} in the given time layer. Computational work is repeated according to this scheme, until the following condition is satisfied:

$$\sqrt{\left(\sum_{i=1}^{Nx}\sum_{i=1}^{Ny}(P_{ij}^{n+1}-P_{ij}^{n})^2\right)} < \varepsilon \tag{4}$$

Using the algorithm described above, a program for solving the problem of oil displacement has been developed. The next task is to optimize this program. Optimization is considered in three ways:

1. Optimization by the compiler
2. Vectorization
3. Other details

0.2. **Optimization by the compiler.** For the first level of optimization, you must select the compiler and use the optimization keys. Currently, there are many different compilers. In this work, Intel C++ Compiler (hereinafter icc) and GNU Compiler Collection compilers (hereinafter gcc) were chosen for the measurements. The calculations were made on an Intel® Xeon® Processor E5-2603. It is known that icc optimizes the program well. Intel's compiler is a commercial project, and it generates good executable code for Intel processors. GCC is a widely known and non-commercial open source compiler. Each compiler has its own set of optimization. We can connect them using the keys -O1, -O2, -O3. Each level -O optimization includes various ways of optimizing the code. For example, in the -O2 and -O3 keys, automatic code vectorization is enabled using SIMD extensions. The -O2 key does not include runtime optimization due to the increase in the length of the executable code. The -O3 key includes cycle loops and function substitution, and this key can reduce the execution time of the program but increase the code size [4,5].

In the first level, optimization is also considered using statistics. When the program is compiled with keys, the compiler cannot optimize based on the data with which the program will work. The Profile-guided optimization (PGO) technology makes it possible to solve this optimization issue. The algorithm for using PGO is very similar to both compilers.

1. Compile the program using the -fprofile-generate key in gcc and -prof-gen in icc.
2. Run the program several times. A file with the extension .gcda in gcc appears, and in icc with the extension .dyn. These files collect and store statistics.
3. We compile the program using the -fprofile-use key in gcc and -prof-use in icc. We get an optimized program.

0.3. **Vectorization.** In modern microprocessors there are vector extensions such as MMX, SSE, AVX, etc. A scalar command can process only one unit of data per clock cycle, and a vector instruction can process a vector of length n. In SSE extensions, the vector length is 128 bits, which means that one command can simultaneously process a vector of four elements of the float type. The length of the vector in the AVX extension is 256 bits, into which you can put 4 elements of a double type or 8 elements of the float type. Code vectorization can be implemented automatically, semi-automatically or manually. Automatic vectorization is realized with special keys at compile time. In icc, auto-vectorization is included in the -O2 key, and in gcc in the -O3 key. But the compiler may not vectorize sections of code that are interdependent or work with integer mathematics, because the final results may not be accurate from the order of the operation [6]. The programmer can explicitly indicate the parts of the code that need to be vectorized. Such actions can be performed using the SIMD directive or in OpenMP 4.0 using the #pragma omp simd pragma. This method of vectorization is called semiautomatic [7,8].

Manual code vectorization can be performed using assembler inserts, C ++ classes SSE in icc or SSE Intrinsics. For the vectorization of the program, SSE Intrinsics instructions were used in this paper. For the vectorization of the program, SSE Intrinsics instructions were used in this paper. The main cycle of calculating the pressure by formula (3), the cycle of finding the matrices M_x and M_y, the cycle for checking the condition (4), the cycle of copying the boundaries by the symmetry condition was vectorized. The AVX vector expansion is installed in the processor in

which the calculations were made. Using data types and commands, the code for this extension was vectorized.

0.4. **Other details.** Optimization of the program can be done for the execution time or for the code size. This work is devoted to reducing the execution time of the program. In the original program, coefficients matrices k_x and k_y were calculated, which was spent a lot of time. The calculation of the matrices M_x and M_y was optimized without superfluous calculation of k_x and k_y and substitution of functions f_1 and f_2. In the formula (3), $M_{i+\frac{1}{2}j}$ means $(M_{ij} + M_{i+1j})/2$. By mathematical means, you can get rid of all divisions by two, because the division takes a lot of processor time, and the reduction in the code of such operations gives acceleration. Also, in the original program, boundary copying occurred in two cycles, one for the upper and lower bounds, the second for the lateral boundaries. And in the optimized version due to the vectors, the lateral boundaries are immediately calculated from the mirror data, and only the upper and lower boundaries are copied. The functions f_1 and f_2 mean $S^{3.5}$ and $(1-S)^{3.5}$, respectively. But in Intrinsics there is no command to find a degree. Initially, in the vector code, 7 degrees of the number were found and brought into the square root, but optimized in such a way that the square root of the number is in another variable, and in the other variable we find 3 degrees of a given number, and multiplying two variables gives 3.5 degrees. It is assumed that the calculating of the two above-described variables runs in parallel. When developing programs, you must always take into account the time of access to memory. When accessed, memory is loaded by blocks, and these blocks fall into the fast cache memory. If each time to call different blocks and each time to load them into the cache memory, the execution time of the program can increase many times. To work around this problem, it is advisable to work with consecutive or close memory cells. For example, it is better to treat arrays as they are stored in memory, that is, line by line.

1. Results and discussion

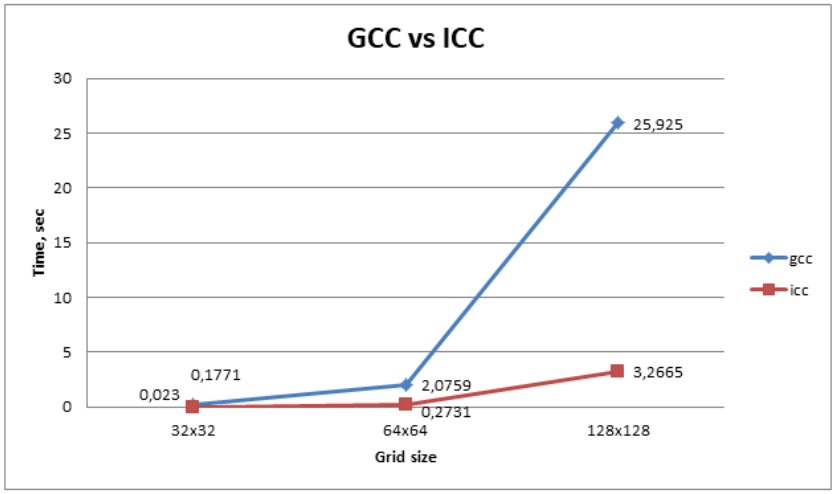

Figure 2 – The execution time of programs without optimization keys. Axis Oy - time of execution, axis Ox - dimensions of the reservoir

The second figure compares the execution time of the program compiled in the ICC and GCC compilers. As you can see, the ICC program time is much less than the GCC program. This is because by default ICC compiles with the -O2 key, and GCC without optimization. And in the third figure, the execution time of programs without optimization. In Figures 4,5 and 6 you can see the execution time with optimization keys. From these graphs we can see that the Intel compiler optimizes the program well. But the advantage of ICC can also be explained by the processor on which the calculations were made. In Intel processors it is better to use the ICC compiler, as well as in AMD processors it is better to use the compiler from AMD. Because the manufacturer is better aware of the advantages of its devices. In Figures 7 and 8 we can see the benefits of

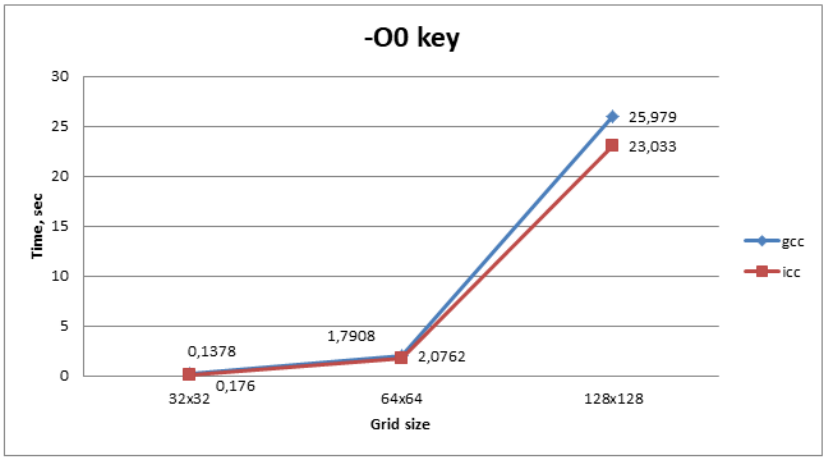

Figure 3 – The execution time of programs with the -00 key. Axis Oy - time of execution, axis Ox - dimensions of the reservoir

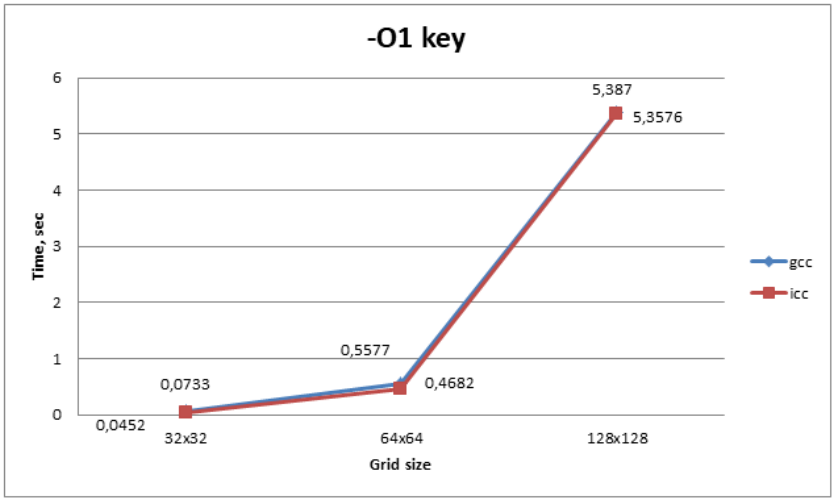

Figure 4 – The execution time of programs with the -O1 key. Axis Oy - time of execution, axis Ox - dimensions of the reservoir

Table 1 – Acceleration of optimization

Grid size	The execution time of the initial program (T_1), sec	Run time of the optimized program (T_2), sec	T_1/ T_2
32x32	0,19735	0,0102	19,34804
64x64	2,3026	0,1491	15,44333
128x128	28,6307	2,007	14,26542
256x256	219,0439	23,869	9,17692

using PGO. The compilation was done with the -O2 key. As you can see, PGO in GCC gives an acceleration of 7-30%. In the ninth figure, you can see a comparison of a scalar program with a vector program. The compilation was performed with the -O2 key. The graph shows that the vectorization yielded approximately 30-50% of the productivity gain. Ideally, the vectorization should speed up the program four times, because four times less operations will be performed. But there is no such acceleration in the graph, because the auto-vectorization is already included in the -O2 key in the Intel compiler. Since the compiler can not vectorize some parts of the code, manual vectorization

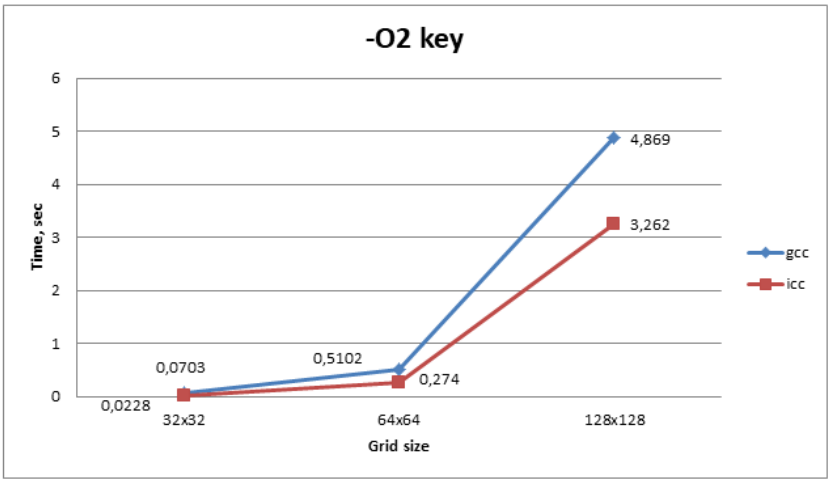

Figure 5 – Time for executing programs with the -O2 key. Axis Oy - time of execution, axis Ox - dimensions of the reservoir

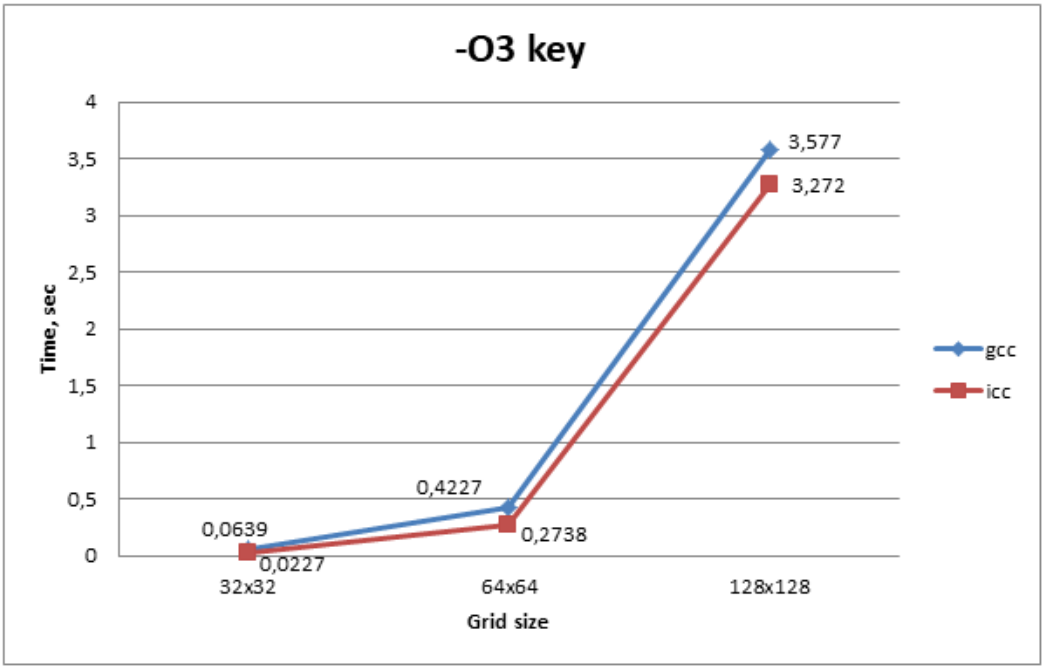

Figure 6 – Time for executing programs with the –O3 key. Axis Oy - time of execution, axis Ox - dimensions of the reservoir

speeds up better. If you compare the execution time of the vectorized version of the program and the scalar version compiled with the -O1 key, you can see a gain of 60-73%. At best, almost four times as mentioned above. And at the end, in the table you can see the comparison of the execution time of the initial program without any optimization methods and the program in which we used three levels of optimization. At best, we got almost 20 times the acceleration. Based on these indications, we can confidently say that the above three levels of optimization should be taken into account when developing programs.

Conclusion. This work was devoted to the study of methods for optimizing programs. To study this methods the program for solving the problem of oil displacement was chosen. A sequential algorithm for solving this problem was compiled, a program was developed, and the code was optimized with three levels. In the first level, optimizing compilers with keys were studied. In the second level, we wrote a vectorized version of the program that accelerated the scalar version. In the third level,

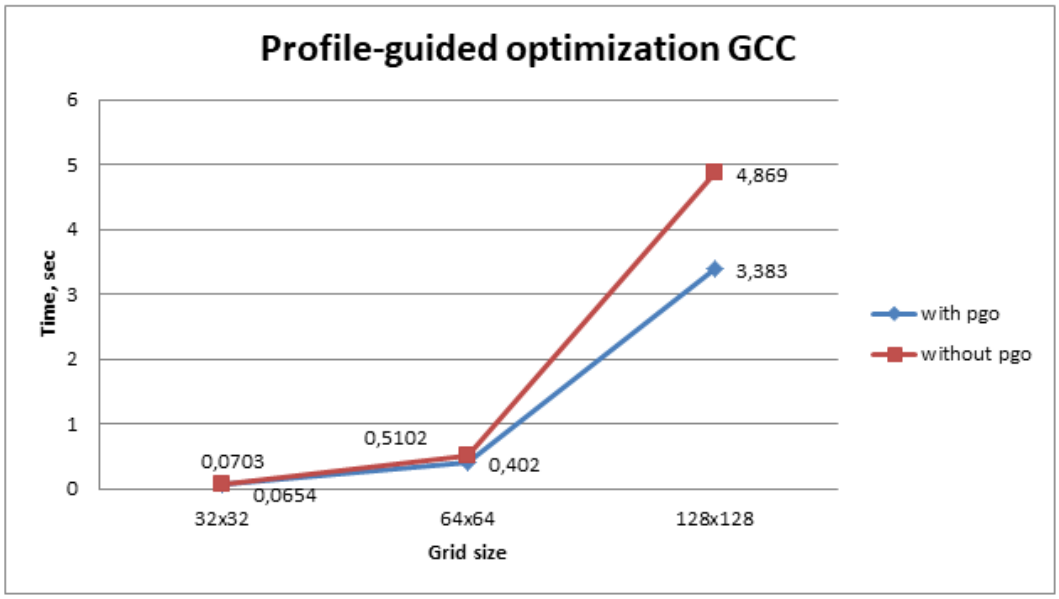

Figure 7 – Optimization using statistics on the GCC compiler. Axis Oy - time of execution, axis Ox - dimensions of the reservoir

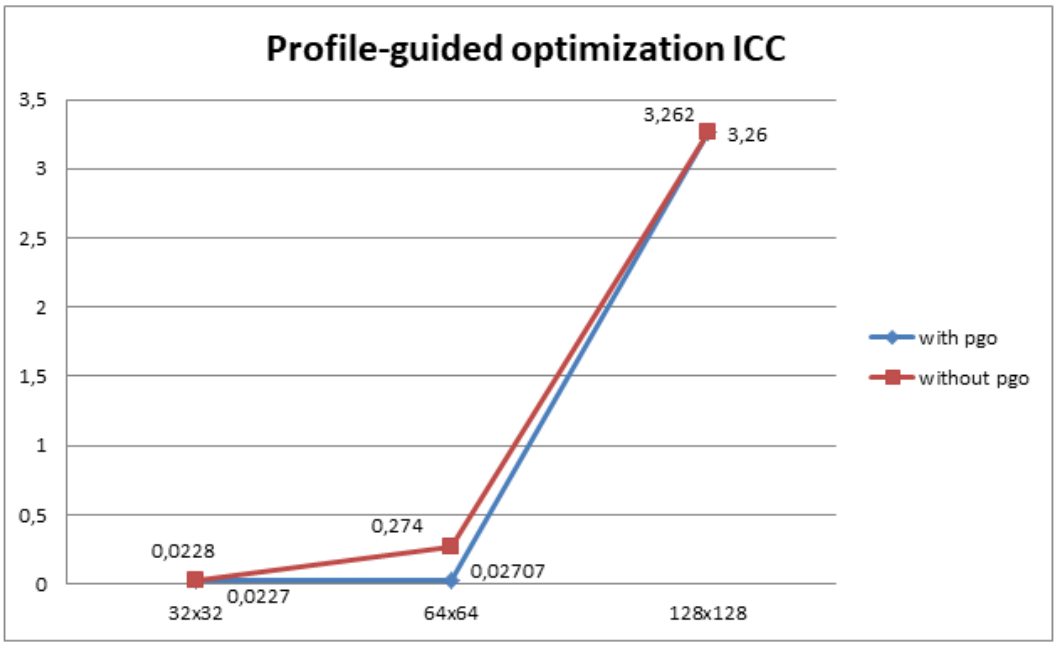

Figure 8 – Optimization using statistics on the Intel compiler. Axis Oy - time of execution, axis Ox - dimensions of the reservoir

other optimization details were considered. Using the methods studied, the initial program without optimization was accelerated by 89-95%. The set goals were achieved, and came to the conclusion that it is possible and necessary to speed up the execution time of programs using optimization methods.

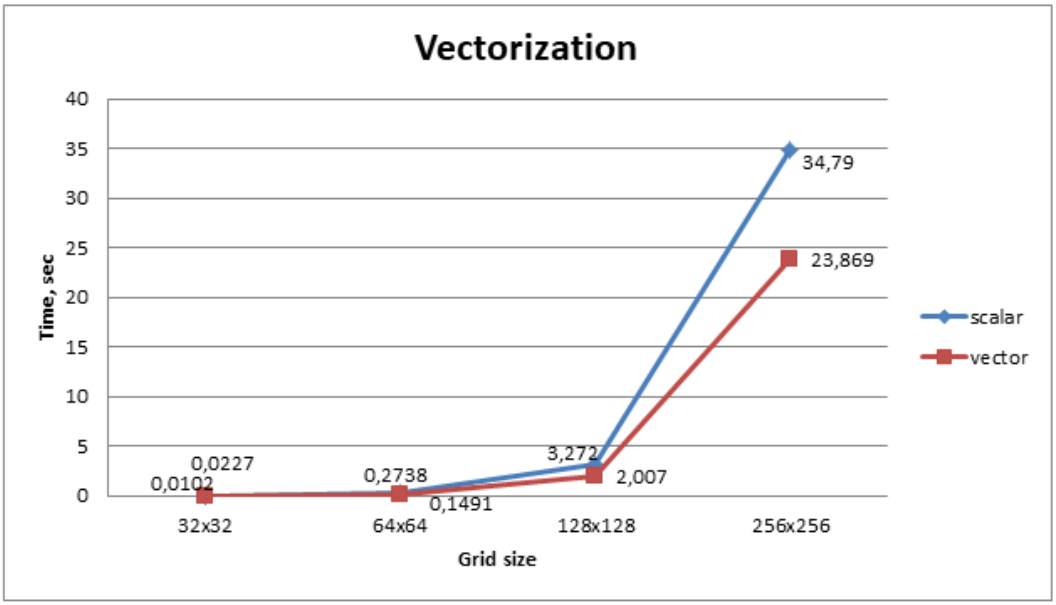

Figure 9 – Execution time after vectorization. Axis Oy - time of execution, axis Ox - dimensions of the reservoir

References

1 Ларри Л. Основы методов увеличения нефтеотдачи. – Остин, 2005. – 449 с.
2 Данаев Н.Т., Корсакова Н.К., Пеньковский В.И. Массоперенос в прискважинной зоне и электромагнитный каротаж пластов. – Алматы, 2005. – 180 с.
3 Самарский А.А., Гулин А.В. Численные методы / учеб. пособие для вузов. – М.: Наука, 1989. – 432 с.
4 Командные Опции GNU CC.++ http://www.opennet.ru/docs/RUS/gcc/gcc1-2. URL:html#ss2.6 [Электронный ресурс](дата обращения: 20.03.2019).
5 Оптимизируем шаг за шагом с компилятором Intel C++ URL:https: //habra-habr.ru/company/intel/blog/256251/ [Электронды ресурс]
6 Чинин Г. Векторизация программ. Теория, методы, реализация. Сборник статей. –М.: Антология, 1991. – 272 с.
7 Антонов А.С. Параллельное программировние с использованием технологии OpenMP. – М.: Изд-во МГУ, 2009. - 77 с.
8 Открытая книга по технологии OpenMP 4.0.URL:http://dautovri.github.io/OpenMP-Book/ [Электронный ресурс](дата обращения: 20.03.2019).
9 Спецкурс "Эффективное программирование современных микропроцессоров и мультипроцессоров". URL: http://ssd.sscc.ru/ru/chair/nsu/programming [Электронный ресурс](дата обращения: 20.03.2019).

Н.М. Қасымбек [1] , М.Б. Мұстафин [1] , Т.С. Иманкулов [1] , Д.Ж. Ахмед-Заки [2]

[1] *Әл-Фараби атындағы Қазақ ұлттық Университеті, Алматы, Қазақстан*
[2] *Халықаралық бизнес университеті, Алматы , Қазақстан*

Мұнай ығыстыру есебін шешуге арналған бағдарламасын оңтайландыру

Аңдатпа: Бұл мақалада мұнай ығыстыру есебін шешу бағдарламасының оңтайландыру әдістері талқыланады. Параллельді архитектураларды пайдаланбай, бағдарламаның орындалу уақытын бір процессор үшін жеделдету қарастырылады. Мұнайды ығыстыру есебі ретінде Якоби әдісімен шешілген екі өлшемді Баклей-Леверетт моделі таңдалды. Бағдарламаны оңтайландыру үш деңгейде жүзеге асырылады, оның негізгісі - векторизация. Оңтайландырудың әрбір деңгейінен кейін бағдарламаның орындалу уақытына талдау жүргізіледі және тізбекті бағдарламамен салыстырылады. Оңтайландырудың барлық үш деңгейі шамамен 89-95% жылдамдықты көрсетті.

Түйін сөздер: мұнай ығыстыру, оңтайландыру, оңтайландыру кілттері, векторизация, SIMD кеңейтілімдері, SSE Intrinsics.

Н.М. Касымбек [1] , М.Б. Мустафин [1] , Т.С. Иманкулов [1] , Д.Ж. Ахмед-Заки [2]

[1] *Казахский национальный университет имени Аль-Фараби, Алматы, Казахстан*
[2] *Университет международного бизнеса, Алматы, Казахстан*

Оптимизация программы для решения задачи вытеснения нефти

Аннотация: В данной статье рассматриваются методы оптимизации программы для решения задачи вытеснения нефти. Рассматривается ускорение времени выполнения программы на одном процессоре без использования параллельных архитектур. В качестве задачи вытеснения нефти выбрана двумерная модель Баклея-Леверетта, которая решалась методом Якоби. Оптимизация программы выполняется тремя уровнями, основной из которых является векторизация. После каждого уровня оптимизации производится анализ времени выполнения программы и сравнение с последовательной программой. Все три уровня оптимизации показали ускорение примерно в 89-95%.

Ключевые слова: вытеснение нефти, оптимизация, ключи оптимизации, векторизация, расширения SIMD, SSE Intrinsics.

References

1 Larri L. Osnovy metodov uvelichenija nefteotdachi [Basics of enhanced oil recovery] (Ostin, 2005).[in Russian]
2 Danaev N.T., Korsakova N.K., Pen'kovskij V.I. Massoperenos v priskvazhinnoj zone i jelektromagnitnyj karotazh plastov[Mass transfer in the near-well zone and electromagnetic logging] (Almaty, Qazaq universiteti,2005). [in Russian]
3 Samarskij A.A., Gulin A.V. Chislennye metody. Ucheb posobie dlja vuzov [Numerical methods]:Universitty textook(Nauka, Moscow, 1989). [in Russian]
4 Komandnye Opcii GNU CC. Available at: http://www.opennet.ru/docs/RUS/gcc/gcc1-2.html#ss2.6. [in Russian]
5 Optimiziruem shag za shagom s kompiljatorom Intel C++ [Optimize step by step with the Intel C ++ compiler]. Available at: https://habrahabr.ru/company/intel/blog/256251/. [in Russian]
6 Chinin G. Vektorizacija programm [Vectorization of programs]. Teorija, metody, realizacija [Theory, methods, implementation] (Antologija, Moscow, 1991, p. 272). [in Russian]
7 Antonov A.S. Parallel'noe programmirovnie s ispol'zovaniem tehnologii OpenMP [Parallel programming using OpenMP technology] (MGU, Moscow, 2009). [in Russian]
8 Otkrytaja kniga po tehnologii OpenMP 4.0 [Open book on technology OpenMP 4.0]. Available at: http://dautovri.github.io/OpenMP-Book/. [in Russian]
9 Speckurs "Jeffektivnoe programmirovanie sovremennyh mikroprocessorov i mul'tiprocessorov" [Efficient programming of modern microprocessors and multiprocessors]. Available at: http://ssd.sscc.ru/ru/chair/nsu/programming. [in Russian]

Сведения об авторах:

Касымбек Н.М. - Магистрант Казахского национального университета имени Аль-Фараби, Алматы, Казахстан.
Мустафин М.Б. - Магистрант Казахского национального университета имени Аль-Фараби, Алматы, Казахстан.
Иманкулов Т.С. - и.о. доцента кафедры Информатики, Казахский национальный университет имени Аль-Фараби, Алматы, Казахстан.
Ахмед-Заки Д.Ж. - Президент Университета международного бизнеса, Алматы, Казахстан.
Kassymbek N.M. - Master student of the Al-Farabi Kazakh National University, Almaty, Kazakhstan.
Mustafin M.B. - Master student of the Al-Farabi Kazakh National University, Almaty, Kazakhstan.
Imankulov T.S. - Associate Professor, Department of Computer Science, Al-Farabi Kazakh National University, Almaty, Kazakhstan.
Akhmed-Zaki D.Zh. - President of University of International Business, Almaty, Kazakhstan.

Поступила в редакцию 16.10.2018