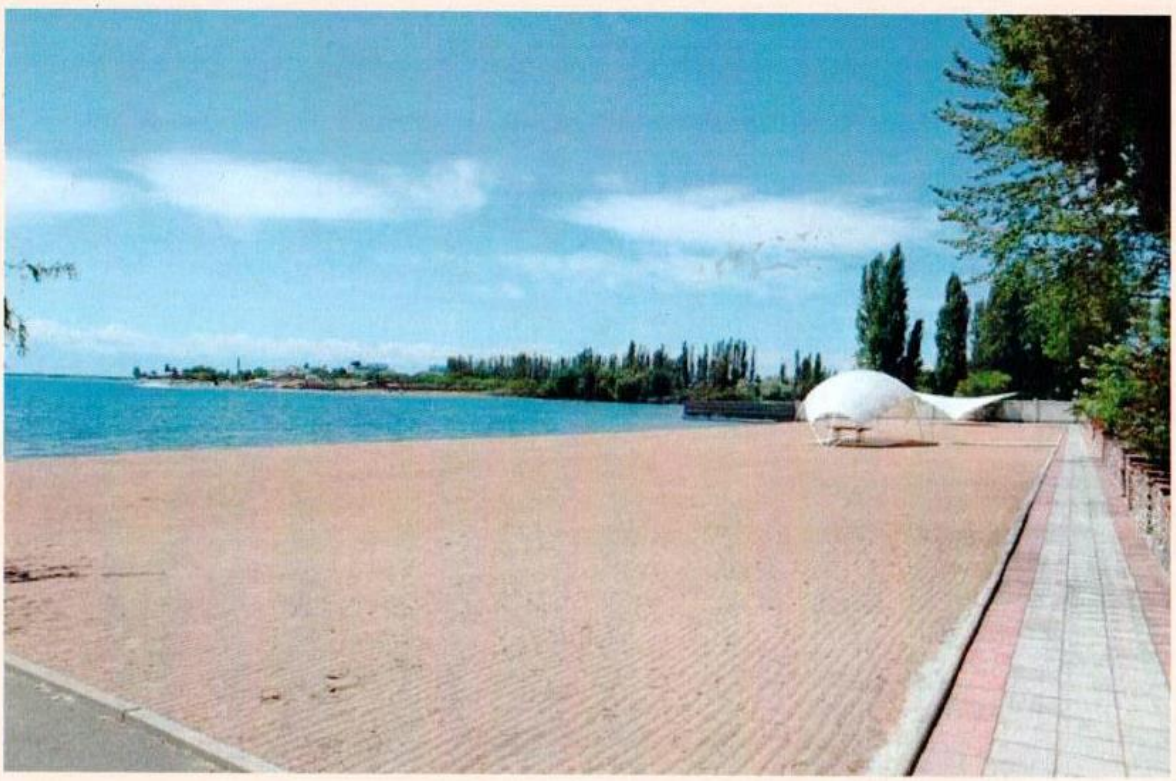

МАТЕРИАЛЫ

XIV Международной Азиатской школы-семинара «ПРОБЛЕМЫ ОПТИМИЗАЦИИ СЛОЖНЫХ СИСТЕМ»

20 - 31 июля 2018 года

ЧАСТЬ 1

Кыргызская Республика
оз. Иссык-Куль
пансионат «Отель Евразия»

Алматы 2018

Институт информационных и вычислительных технологий
МОН РК (Республика Казахстан, г. Алматы)

Институт вычислительной математики и математической
геофизики СО РАН (Россия, г. Новосибирск)

Новосибирский государственный университет
(Россия, г. Новосибирск)

Институт математики НАН КР
(Кыргызская Республика, г. Бишкек)

Институт автоматики и информационных технологий НАН КР
(Кыргызская Республика, г. Бишкек)

ОсОО "Силк Роад энд Кыргыз Гайдеде Трипс"
(Кыргызская Республика, г. Бишкек)

МАТЕРИАЛЫ
XIV Международной Азиатской школы-семинара
«ПРОБЛЕМЫ ОПТИМИЗАЦИИ СЛОЖНЫХ СИСТЕМ»

20 июля – 31 июля 2018 г.

Часть 1

Кыргызская Республика
оз. Иссык-Куль
пансионат «Отель Евразия»

с. Кара-Ой

Алматы
2018

наук Республики Казахстан. Серия физико-математическая. - Алматы: Гылым, 2005. - № 3. - С. 84-89.

6. Vergili I., Yücel M. D. Avalanche and Bit Independence Properties for the Ensembles of Randomly Cho-sen S-Boxes // Turk J Elec Engin.. - № 2. С. 137–145. -2001

Нурсулу Алдажаровна Капалова – к.т.н., ВНС Института информационных и вычислительных технологий МОН РК; 050010 Алматы; e-mail: nkapalova@mail.ru;

Дилмуханбет Самуратович Дюсенбаев – инженер-программист Института информационных и вычислительных технологий МОН РК; 050010 Алматы; e-mail: dimash_dds@mail.ru

Кунболат Тилеуханулы Алгазы – магистр, МНС Института информационных и вычислительных технологий МОН РК; 050010 Алматы; e-mail: kunbolat@mail.ru

Армиянбек Хаумен – магистр, инженер-программист Института информационных и вычислительных технологий МОН РК; 050010 Алматы; e-mail: haumen.armanbek@gmail.com

УДК 519.687.1

ОПТИМИЗАЦИЯ ПРОГРАММЫ ДЛЯ РЕШЕНИЯ ЗАДАЧИ ВЫТЕСНЕНИЯ НЕФТИ

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

[1] Казахский национальный университет имени аль-Фараби,
[2] Университет международного бизнеса

***Аннотация**. В данной работе рассматривается оптимизация программы для решения задачи вытеснения нефти и изучаются методы оптимизации. Изучено ускорение времени выполнения программы на одном процессоре, без использования параллельных архитектур. Оптимизация производится тремя уровнями, самый из них векторизация программы. Методом тестирования программы получен и проанализированы результаты оптимизации.*

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

Введение

Численные модели современных задач промышленности и науки работают с очень большими данными. Для вычислений уходит много процессорного времени. Для уменьшения времени вычисления используются параллельные технологии вычисления, как MPI, OpenMP, CUDA и др. Но для параллельных вычислений необходимо потребуется система с несколькими вычислительными процессорами. Но имеет смысл оптимизировать программу для одного процессора. Данная работа посвящена оптимизации программы, используя потенциал современных процессоров.

оптимизации программ значительную роль может играть компилятор, чем лучше компилятор справляется с оптимизацией, тем быстрее может работать ваша программа. Ускорить время работы программ, можно используя векторные расширения процессоров, таких как SSE, AVX и другие версии. А также есть и другие методы оптимизации, которых можно реализовать исходя от работы и цели программы. Так как время исполнения одна из важных величин, у оптимизации значительная роль при разработке программ.

1. Постановка задачи и метод решения

Для решения задачи вытеснения нефти была выбрана модель Баклея-Леверетта [1]. Рассмотрим случай когда через нагнетающую скважину за определенное время проходит определенное количество воды. В нагнетающие и добывающие скважины даются давления P_{inj} и P_{prod} $(P_{inj} > P_{prod})$. Вода, протекающая через нагнетающую скважину вытесняет нефть в пласте и нефть попадает в добывающую скважину. Нужно разработать компьютерную модель массообменных процессов в пласте. Ниже приведена схема пласта:

Математическая модель двухфазовой фильтрации состоит из уравнении баланса воды и нефти. Через границы $\partial\Omega$ для области Ω уравнение системы будет в следующем виде [2]:

$$m\frac{\partial s_1}{\partial t} + div(\vec{v}_1) = 0 \quad (1)$$

$$m\frac{\partial s_2}{\partial t} + div(\vec{v}_2) = 0 \quad (2)$$

$$\vec{v}_1 = -k\frac{f_1}{\mu_1}\nabla P \quad (3)$$

$$\vec{v}_2 = -k\frac{f_2}{\mu_2}\nabla P \quad (4)$$

Здесь, m – пористость пласта, s_1, s_2 – насыщенность воды и нефти, $0 \le s_1, s_2 \le 1$, $s_1 + s_2 = 1$, $\vec{v}_1, \vec{v}_2$ – скорости, k – абсолютная проницаемость, f_1, f_2 – относительные фазовые проницаемости, μ_1, μ_2 – вязкость, P – давление. Таким образом, нужно найти функцию P, давление, используя следующие начальные условия

$$P|_{t=0} = P_{пл} \quad (5)$$

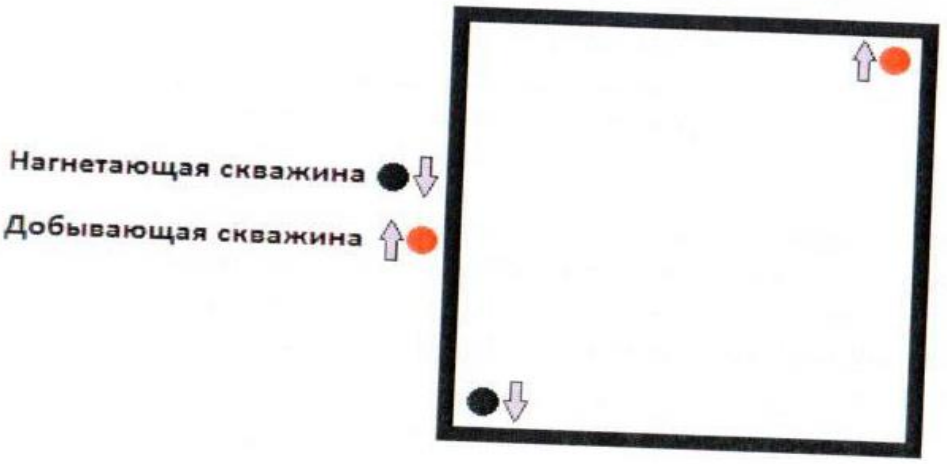

Рис. 1. Схема скважины

За граничные условия было взято условие симметричности:

$$\left.\frac{\partial P}{\partial x}\right|_{x=0} = \left.\frac{\partial P}{\partial x}\right|_{x=1} = 0 \quad (6)$$

$$\left.\frac{\partial P}{\partial y}\right|_{y=0} = \left.\frac{\partial P}{\partial y}\right|_{y=1} = 0 \quad (7)$$

Дифференциальный вид уравнения:

$$m\frac{\partial s_1}{\partial t} + \frac{\partial}{\partial x}\left(-k_x\frac{f_1}{\mu_1}\frac{\partial P}{\partial x}\right) + \frac{\partial}{\partial y}\left(-k_y\frac{f_1}{\mu_1}\frac{\partial P}{\partial y}\right) = 0 \quad (8)$$

$$m\frac{\partial s_2}{\partial t} + \frac{\partial}{\partial x}\left(-k_x\frac{f_2}{\mu_2}\frac{\partial P}{\partial x}\right) + \frac{\partial}{\partial y}\left(-k_y\frac{f_2}{\mu_2}\frac{\partial P}{\partial y}\right) = 0 \quad (9)$$

$$\frac{\partial}{\partial x}\left(M_x\frac{\partial P}{\partial x}\right) + \frac{\partial}{\partial y}\left(M_y\frac{\partial P}{\partial y}\right) = 0 \quad (10)$$

Для компьютерных расчетов уравнение вытеснения нефти (10) было приведено в конечно-разностный вид и используя итерационный метод получили следующую формулу [3]:

$$\left((M_x)_{i+\frac{1}{2}j}\frac{P^n_{i+1j}-P^{n+1}_{ij}}{h_1^2} - (M_x)_{i-\frac{1}{2}j}\frac{P^{n+1}_{ij}-P^n_{i-1j}}{h_1^2} + (M_y)_{ij+\frac{1}{2}}\frac{P^n_{ij+1}-P^{n+1}_{ij}}{h_2^2} - (M_y)_{ij-\frac{1}{2}}\frac{P^{n+1}_{ij}-P^n_{ij-1}}{h_2^2}\right) = 0 \quad (11)$$

Из этого уравнения можно получить следующую формулу расчета давления в каждой точке пласта:

$$P^{n+1}_{ij} = \frac{(M_x)_{i+\frac{1}{2}j}P^n_{i+1j} + (M_x)_{i-\frac{1}{2}j}P^n_{i-1j} + (M_y)_{ij+\frac{1}{2}}P^n_{ij+1} + (M_y)_{ij-\frac{1}{2}}P^n_{ij-1}}{(M_x)_{i+\frac{1}{2}j} + (M_x)_{i-\frac{1}{2}j} + (M_y)_{ij+\frac{1}{2}} + (M_y)_{ij-\frac{1}{2}}} \quad (12)$$

Используя данные в n-ном временном слое находим давления P^{n+1}_{ij} в данном временном слое. Вычислительные работы повторяются по этой схеме, пока следующее условие не будет удовлетворено:

$$\sqrt{\left(\sum_{i=1}^{Nx}\sum_{i=1}^{Ny}(P^{n+1}_{ij} - P^n_{ij})^2\right)} < \varepsilon \quad (13)$$

Используя выше описанный алгоритм разработана программа решения задачи вытеснения нефти. Следующая задача оптимизировать эту программу. Оптимизация рассмотрена тремя путями:

1. Оптимизация компилятором
2. Векторизация
3. Другие подробности

2. Оптимизация компилятором

Для первого уровня оптимизации необходим выбор компилятора и использование ключей оптимизации. В настоящее время есть много разных компиляторов. В данной работе для измерений были выбраны компиляторы Intel C++ Compiler(далее icc) и GNU Compiler Collection(далее gcc). Расчеты были произведены на процессоре Intel® Xeon® Processor E5-2603. Известно, что icc хорошо оптимизирует программу. Компилятор Intel коммерческий проект, и он генерирует хороший исполняемый код для процессоров Intel. GCC широко известный и некоммерческий компилятор с открытым кодом. У каждых компиляторов есть свой набор оптимизации. Их мы можем подключить, используя ключи –O1, -O2, -O3. Каждый уровень -O оптимизации включает в себя разные способы оптимизации кода. Например, в ключах –O2 и –O3 включается автоматическая векторизация кода используя расширения SIMD. Ключ –O2 не включают оптимизацию по времени выполнения за счет увеличения длины исполняемого кода. Ключ –O3 включает раскрутку циклов и подстановку функций, и этот ключ может уменьшить время выполнения программы, но увеличить размер кода [4,5].

На первом уровне также рассмотрена оптимизация используя статистику. Когда программа компилируется с ключами, компилятор не может оптимизировать исходя от данных с которыми будет работать программа. Технология Profile-guided optimization (далее PGO) дает возможность решить этот вопрос оптимизации.

3. Векторизация

В современных микропроцессорах имеются векторные расширения, такие как MMX, SSE, AVX и т.д. Скалярная команда за один такт процессорного времени может обработать только единицу данных, а векторная команда может обработать вектор длины n. В расширениях SSE длина вектора равна 128 бит, это означает, что одна команда может обработать одновременно вектор из четырех элементов типа float. Длина вектора в расширении AVX равна 256 битам, в которые поместятся 4 элемента типа double или 8 элементов типа float. Векторизацию кода можно реализовать автоматическими, полуавтоматическими или ручными путями. Автоматическая векторизация реализуется специальными ключами во время компиляции. В icc автовекторизация включена в ключ –O2, а в gcc в ключ –O3. Но компилятор может не векторизовать участки кода, которые взаимозависимы или работают с целочисленной математикой, потому что от порядка выполнения операции конечные результаты могут быть неточными [6]. Программист может явно указать участки кода, которые должны быть векторизованы. Такие действия могут быть выполнены с помощью директивы SIMD или же в OpenMP 4.0, используя прагму #pragma omp simd. Такой метод векторизации называется полуавтоматическим [7,8].

Ручную векторизацию кода можно выполнить, используя ассемблерные вставки, C++ классы SSE в icc или же SSE Intrinsics. Для векторизации программы в данной работе были использованы инструкции SSE Intrinsics. Intrinsics – набор функций и типов данных, поддерживаемых компилятором, для предоставления доступа к векторным инструкциям. Были векторизованы основной цикл расчета давления по формуле (12), цикл нахождения матриц M_x и M_y, цикл на проверку условия (13), цикл копирования границ по условию симметричности. В процессоре, в котором производились расчёты, установлено векторное расширение AVX. Используя типы данных и команды, векторизовали код для данного расширения.

4. Другие подробности

Оптимизацию программы можно сделать по времени выполнения или по размеру кода. Данная работа посвящена уменьшению времени исполнения программы. В исходной программе рассчитывались матрицы коэффициентов k_x и k_y, на что тратилось время. Был оптимизирован расчет матриц M_x и M_y без лишнего расчета k_x и k_y и подстановкой функций f_1 и f_2. В формуле (12) $M_{i+\frac{1}{2}j}$ означает $(M_{ij} + M_{i+1j})/2$. Математическим путём можно избавиться от всех делений на два, потому что на деление уходит много процессорного времени, и уменьшение в коде таких операций дает ускорение. Также, в исходной программе копирование границ происходило в двух циклах, один для верхнего и нижнего границ, второй для боковых границ. А в оптимизированной версии благодаря векторам боковые границы сразу вычисляются по зеркальным данным, и копируются только верхние и нижние границы. Функции f_1 и f_2 означают $S_{ij}^{3.5}$ и $(1 - S_{ij})^{3.5}$ соответственно. Но в Intrinsics нет команды нахождения степени. Изначально, в векторном коде находили 7 степень числа и приводили в квадратный корень, но оптимизировали таким образом, что находится квадратный корень числа в другую переменную, а в другой переменной находим 3 степени данного числа, а умножение двух переменных дает 3,5 степени. Предполагается, что нахождение двух вышеописанных переменных проходит параллельно. При разработке программ нужно обязательно учесть время доступа к памяти. При доступе память загружается блоками, и эти блоки попадают в быструю кэш память. Если каждый раз вызывать разные блоки и каждый раз загружать их в кэш память, время исполнения программы может увеличиться в разы. Для обхода этой проблемы советуется работать с последовательными или близкими ячейками памяти. Например, в массивы лучше обращаться так, как они хранятся в памяти, то есть построчно.

5. Результаты

Рис. 2. Время исполнения программ без ключей оптимизации

Рис. 3. Время выполнения программ с ключом –O0

Рис. 4. Время выполнения программ с ключом –O1

Рис. 5. Время выполнения программ с ключом –O2

Рис. 6. Время выполнения программ с ключом –O3

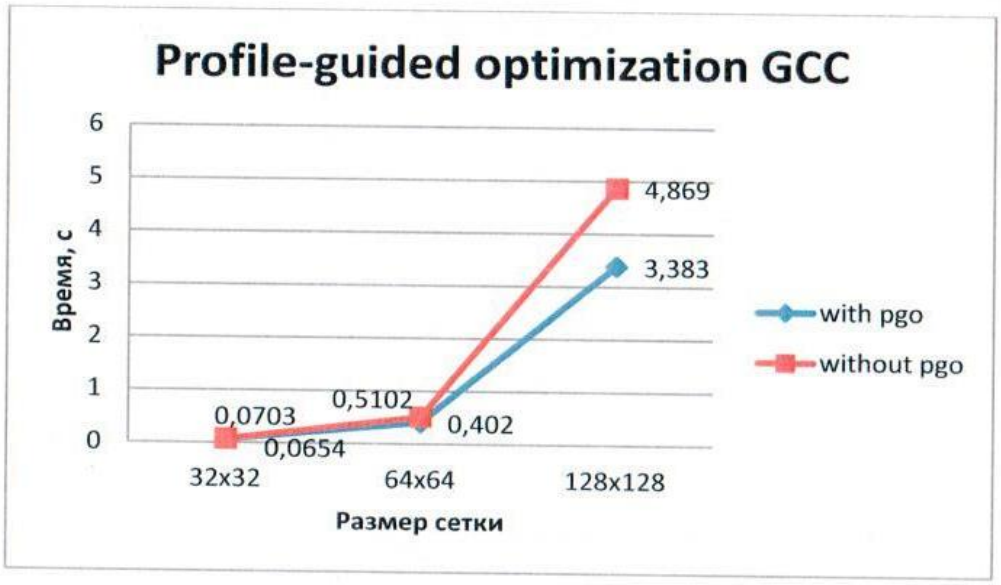

Рис. 7. Оптимизация с использованием статистики на компиляторе GCC

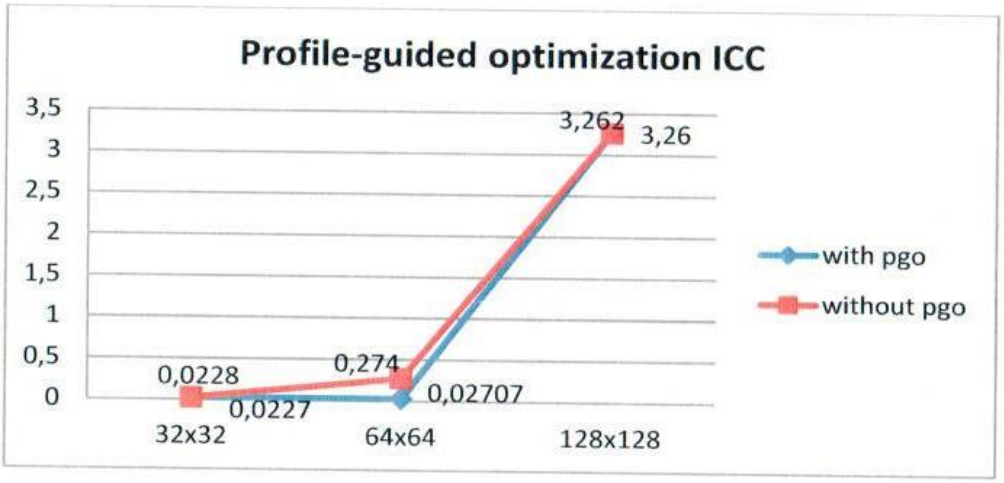

Рис. 8. Оптимизация с использованием статистики на компиляторе Intel

Рис. 9. Время исполнения после векторизации

Таблица 1. Ускорение оптимизации

Размер пласта	Время выполнения начальной программы, с	Время выполнения оптимизированной программы, с	Ускорение
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

6. Заключение

Данная работа была посвящена изучению методов оптимизации программ. Для изучения выбрана программа решения задачи вытеснения нефти. Был составлен последовательный алгоритм решения данной задачи, разработана программа и программу оптимизировали тремя уровнями. В первом уровне изучали оптимизирующие компиляторы с ключами. Во втором уровне написали векторизованную версию программы, которая ускорила скалярную версию. В третьем уровне были рассмотрены другие подробности оптимизации. Используя изученные методы, исходная программа, без оптимизации, была ускорена на 89-95%. Изначально поставленные цели были достигнуты, и пришли к выводу, что можно и нужно ускорить время исполнения программ используя методы оптимизации.

Работа выполнена в рамках грантового финансирования Комитета Науки МОН РК

Список литературы

1. Ларри Лейк. Основы методов увеличения нефтеотдачи. – Остин, 2005. – 449с

2. Данаев Н.Т., Корсакова Н.К., Пеньковский В.И. Массоперенос в прискважинной зоне и электромагнитный каротаж пластов. – Алматы: Қазақ университеті, 2005. – 180 с

3. Самарский А.А. , Гулин А. В. Численные методы / Учеб. Пособие для вузов – М. : Наука, 1989. – 432 с.
4. http://www.opennet.ru/docs/RUS/gcc/gcc1-2.html#ss2.6 [Электронный ресурс] – Командные Опции GNU CC
5. https://habrahabr.ru/company/intel/blog/256251/ [Электронный ресурс] – Оптимизируем шаг за шагом с компилятором Intel C++
6. Г. Чинин Векторизация программ. Теория, методы, реализация. Сборник статей. –М. : Антология, 1991 . – 272 с.
7. Антонов А. С. Параллельное программирование с использованием технологии OpenMP. -М. : Изд-во МГУ , 2009. - 77с.
8. http://dautovri.github.io/OpenMP-Book/ [Электронный ресурс] –Открытая книга по технологии OpenMP 4.0
9. http://ssd.sscc.ru/ru/chair/nsu/programming - Спецкурс «Эффективное программирование современных микропроцессоров и мультипроцессоров», ФИТ НГУ

Нұрислам Мұратбекұлы Қасымбек – магистрант КазНУ имени аль-Фараби; 050000, Алматы, Казахстан; nuryslamqassymbek@gmail.com

Максат Бейбитович Мустафин – магистрант КазНУ имени аль-Фараби; 050000, Алматы, Казахстан; astanaforeverever@gmail.com

Тимур Сакенович Иманкулов – PhD, и.о. доцента КазНУ имени аль-Фараби; 050000, Алматы, Казахстан; e-mail: imankulov.timur@gmail.com

Дархан Жумаканович Ахмед-Заки – д.т.н., профессор Университет международного бизнеса; 050000, Алматы, Казахстан; e-mail: darhan_a@mail.ru

УДК 004.4

МЕТОДЫ И СИСТЕМА ОБЛАЧНОГО ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ

Касьянов В.Н., Касьянова Е.В.

Институт систем информатики им. А. П. Ершова СО РАН, Новосибирск

Доклад посвящен проекту CPPS, выполняемому в ИСИ СО РАН при поддержке Российского научного фонда. Цель проекта — разработка языковых и программных средств, поддерживающих создание, верификацию и отладку архитектурно-независимых параллельных программ и их корректное преобразование в эффективный код параллельных вычислительных систем различных архитектур. В докладе описывается сама создаваемая система CPPS, ее входной язык Cloud Sisal и язык внутреннего представления Cloud Sisal программ.

Ключевые слова: *аннотированное программирование, графовое представление программ, параллельное программирование, функциональное программирование, язык программирования.*