Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра вычислительных технологий и моделирования

Чаплыгин Андрей Викторович

Параллельное моделирование нелинейных уравнений мелкой воды

КУРСОВАЯ РАБОТА

Научный руководитель:
д.ф.-м.н.
Н.А. Дианский

Москва, 2018

Содержание

1 Введение

В работе рассматривается система нелинейных уравнений мелкой воды, являющаяся блоком сигма-модели общей циркуляции океана INMOM (Institute of Numerical Mathematics Ocean Model), развиваемой в ИВМ РАН (Институт Вычислительной Математики) [1, 2]. Модель написана на языке Fortran 90/95. Предыдущая версия модели INMOM использовалась в качества океанического блока климатической модели INMCM (Institute of Numerical Mathematics Climate Model), созданной в ИВМ РАН и участвующей в программе IPCC по прогнозированию изменений климата [22].

Система нелинейных уравнений мелкой воды является неотъемлемой и одной из самых важных подзадач в моделях общей циркуляции океана. Решение этой системы уравнений занимает существенную часть времени, необходимого для решения полной задачи циркуляции океана, как было показано, например в работе [2]. Поэтому возникает вопрос об эффективной параллельной реализации алгоритма решения системы нелинейных уравнений мелкой воды.

В задачах моделирования океана и моделирования цунами особенно актуальна проблема балансировки нагрузки вычислений на процессоры. Чтобы параллельная программа была эффективной, важно равномерно распределять нагрузку на процессоры. Но из-за наличия берегов и островов в акваториях это не всегда просто сделать.

Целями данной работы являются: реализация эффективного параллельного алгоритма решения нелинейных уравнений мелкой воды для использования на массивно-параллельных вычислительных системах, как в качестве блока модели INMOM, так и независимо; реализация метода балансировки нагрузки на процессоры, позволяющий повысить эффективность параллельной программы; моделирование цунами 2011 года в Японии и экстремального шторма на Азовском море в марте 2013, сравнение результатов расчетов по нелинейным и линеаризованным уравнениями мелкой воды

2 Нелинейные уравнения мелкой воды

В произвольной ортогональной системе координат (x, y) с метрическими коэффициентами (r_x, r_y) (коэффициентами Ламе) полная система нелинейных уравнений мелкой воды записывается в виде [3]:

$$\begin{gathered}
\frac{\partial r_x r_y h u}{\partial t} + T_u(u,v) - F_u(u,v) - h r_x r_y l v + r_y h g \frac{\partial \zeta}{\partial x} + r_x r_y \frac{g n^2}{h^{1/3}} u \sqrt{u^2+v^2} = RHS_u \\
\frac{\partial r_x r_y h v}{\partial t} + T_v(u,v) - F_v(u,v) + h r_x r_y l u + r_x h g \frac{\partial \zeta}{\partial y} + r_x r_y \frac{g n^2}{h^{1/3}} v \sqrt{u^2+v^2} = RHS_v \\
\frac{\partial h}{\partial t} + \frac{1}{r_x r_y}\left(\frac{\partial u r_y h}{\partial x} + \frac{\partial v r_x h}{\partial y}\right) = 0
\end{gathered} \tag{1}$$

Где u, v – компоненты скорости; r_x, r_y – метрические коэффициенты; l – параметр Кориолиса; g – ускорение свободного падения; n – коэффициент донного трения, выписанного в форме, предложенной в японской модели TUNAMI [4]; ζ – отклонение уровня моря относительно невозмущенного состояния; $h = H + \zeta$ – эффективная глубина океана; H – глубина океана в состоянии покоя.

Операторы переноса T_u , T_v записываются в дивергентной форме:

$$\begin{aligned} T_u(u,v,h) &= \frac{\partial h r_y u u}{\partial x} + \frac{\partial h r_x v u}{\partial y} - h\left(v\frac{\partial r_y}{\partial x} - u\frac{\partial r_x}{\partial y}\right)v \\ T_v(u,v,h) &= \frac{\partial h r_y u v}{\partial x} + \frac{\partial h r_x v v}{\partial y} + h\left(v\frac{\partial r_y}{\partial x} - u\frac{\partial r_x}{\partial y}\right)u \end{aligned} \tag{2}$$

С вычислительной точки зрения дивергентная форма обладает полезными свойствами: она сохраняет интеграл переносимой величины по замкнутой области при условии непротекания на твердых границах и выполнения уравнения неразрывности; она допускает простую конечноразностную аппроксимацию. Преимущества дивергентной формы записи уравнений можно посмотреть здесь [5].

Операторы вязкости F_u , F_v записываются как дивергенция тензора напряжений:

$$\begin{aligned} F_u(u,v) &= \frac{1}{r_y}\frac{\partial}{\partial x}\left(r_y^2 K D_T h\right) + \frac{1}{r_x}\frac{\partial}{\partial y}\left(r_x^2 K D_S h\right) \\ F_v(u,v) &= -\frac{1}{r_x}\frac{\partial}{\partial y}\left(r_x^2 K D_T h\right) + \frac{1}{r_y}\frac{\partial}{\partial x}\left(r_y^2 K D_S h\right) \end{aligned} \tag{3}$$

Здесь K - коэффициент вязкости, а D_T и D_S - компоненты тензоров напряжений сжатия-растяжения и сдвига соответственно:

$$\begin{aligned} D_T &= \frac{r_y}{r_x}\frac{\partial}{\partial x}\left(\frac{u}{r_y}\right) - \frac{r_x}{r_y}\frac{\partial}{\partial y}\left(\frac{v}{r_x}\right) \\ D_S &= \frac{r_x}{r_y}\frac{\partial}{\partial y}\left(\frac{u}{r_x}\right) + \frac{r_y}{r_x}\frac{\partial}{\partial x}\left(\frac{v}{r_y}\right) \end{aligned} \tag{4}$$

Использование вязкости в форме (3) и (4) позволяет сохранять свойство отсутствия вязкости при "твердом"вращении жидкости.

В общем случае в правых частях RHS_u , RHS_v рассчитываются градиенты атмосферного давления и напряжения трения ветра.

$$\begin{aligned} RHS_u &= P_x + \tau_x^{surf} \\ RHS_v &= P_y + \tau_y^{surf} \end{aligned} \tag{5}$$

Здесь P_x, P_y - градиенты атмосферного давления на поверхности океана, τ^{surf} - напряжение силы ветра на поверхности. В правых частях также могут рассчитываться и приливные силы, рассчитываемые через приливной потенциал.

На боковой поверхности для скорости задаются граничные условия непротекания и свободного скольжения.

Именно в виде (1) - (5) нелинейные уравнения мелкой воды представлены в модели общей циркуляции океана INMOM [1, 2], возникающие при разрешении быстрых баротропных гравитационных волн.

3 Описание численной реализации

3.1 Дискретизация по пространству

При дискретизации по пространству нелинейных уравнений мелкой воды используется сетка в общем случае нерегулярная по долготе и широте. Разобьем область $\{x \in [x_0, x_{max}], \quad y \in [y_0, y_{max}]\}$, куда входит область, на которой рассматривается система уравнений (1), на элементарные ячейки, которые будут иметь форму прямоугольников:

$$\{(x, y) : x_{m-1} < x < x_m, \quad y_{n-1} < y < y_n\}$$

Для решения системы уравнений (1) применяется техника построения разностных аппроксимаций по пространству второго порядка точности на разнесенной 'C'-сетке по классификации Аракавы [6, 7]. На рис. 1 показаны распределения переменных в каждой сеточной ячейке.

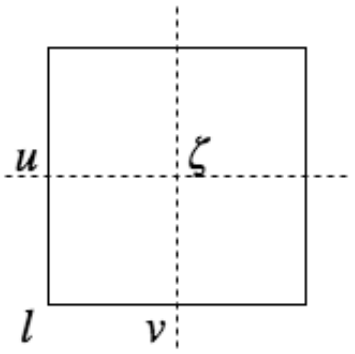

Рис. 1: Распределение переменных на ячейке модельной сетки

Внутри ячейки располагается скалярная величина (ζ). В этих же точках задана топография дна. На соответствующих гранях распределяются потоковые переменные, такие как компоненты вектора скорости (u, v), а также производные скалярных величин по соответствующим направлениям. При этом скорости расположены точно в центре отрезка, соединяющего две соседние точки скалярных величин, что необходимо для получения правильной конечноразностной аппроксимации по пространству [1, 8]. Параметр Кориолиса l определяется в точках вертикальных ребер ячеек.

При построении разностных схем особое место уделяется тому, чтобы сохранялись свойства симметрии разностных аналогов операторов, которые выполняются для дифференциальной задачи. Это позволяет автоматически удовлетворять энергетическим соотношениям в разностной задаче, справедливым для дифференциальной. Методика построения пространственных разностных аппроксимаций хорошо изложена, например, здесь [5, 8].

3.2 Схема по времени

При дискретизации нелинейных уравнений мелкой воды в качестве схемы по времени используется схема 'Чехарда со средней точкой' ('leapfrog'). Для определения решения на шаге $n + 1$ используются решения на шагах n и $n - 1$.

Рассмотрим простейшее уравнение адвекции:

$$\frac{dU}{dt} = F(U) \tag{6}$$

Применяя численную схему по времени 'Чехарда со средней точкой', получим следующее:

$$\frac{U^{n+1} - U^{n-1}}{2\tau} = F(U^n) \tag{7}$$

У такой схемы по времени есть основной недостаток: имеется возможность расчленения решения по временным шагам, т.е. когда развиваются два несвязных расщепленных решения, чередующихся на каждом шаге [5]. Для того, чтобы избежать такого эффекта, на каждом временном шаге n делается фильтрация [9]:

$$U^s = U^n + \frac{a}{2}(U^{n+1} - 2U^n + U^{n-1}) \tag{8}$$

Затем, при переходе на следующий шаг по времени, решение U^s присваивается U^{n-1}, а решение U^{n+1} присваивается U^n. Параметр для фильтрации часто выбирают $a = 0.05$ [9].

Для донного трения в системе уравнений (1) используется неявная схема Эйлера по времени, что повышает устойчивость численного алгоритма.

4 Описание параллельной реализации

4.1 Метод декомпозиции области

В качестве основной идеи параллельного алгоритма решения нелинейных уравнений мелкой воды (1) используется метод декомпозиции области. Суть его в следующем: расчетная область разбивается на подобласти, которые ставятся в соответствие каждому процессору. Процессор проводит вычисления только на своей подобласти и если ему понадобятся данные с соседней подобласти, то осуществляется обмен данными между процессорами.

В качестве базового алгоритма разбиения расчетной области используется равномерное разбиение на прямоугольные подобласти (см. рис. 2).

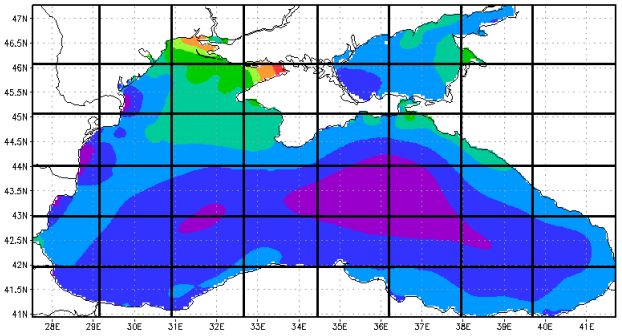

Рис. 2: Метод декомпозиции области: равномерное разбиение на прямоугольные подобласти

Рассмотрим расчетную область, состоящую из множества точек $\{(i,j) : 1 \le i \le n_x, 1 \le j \le n_y\}$, где n_x - количество точек по горизонтали и n_y - количество точек по вертикали. Пусть используется p процессоров, образующих сетку $p = p_x \times p_y$, где p_x - количество процессоров по горизонтали и p_y - количество процессоров по вертикали. Тогда при равномерном разбиении, каждому процессору с координатами (p_1, p_2) ставится в соответствие прямоугольная подобласть из множества точек $\{(i,j) : n_{x,start} \le i \le n_{x,end}, n_{y,start} \le j \le n_{y,end}\}$, где:

$$
\begin{aligned}
n_{x,start} &= \lceil \frac{n_x}{p_x} \rceil (p_1 - 1) + 1 \\
n_{x,end} &= \lceil \frac{n_x}{p_x} \rceil (p_1) \\
n_{y,start} &= \lceil \frac{n_y}{p_y} \rceil (p_2 - 1) + 1 \\
n_{y,end} &= \lceil \frac{n_y}{p_y} \rceil (p_2)
\end{aligned}
\tag{9}
$$

В качестве основного средства распараллеливания используется технология MPI. Так же возможно использование гибридного подхода MPI + OpenMP. В этом случае происходит декомпозиция области и на каждой подобласти используется технология распараллеливания OpenMP, обмен данными между процессорами происходит с помощью технологии MPI.

Реализованы два параллельных алгоритма: HCNC (High-Communication No-Extra Computation), который использует большое количество пересылок между процессорами, но не использует лишние вычисления и LCHC (Low-Communication High-Extra Computation), который уменьшает количество пересылок за счет увеличения внерасчетной области и лишних вычислений [10]. Более детально об этих алгоритмах будет рассказано далее.

4.2 Алгоритмы HCNC и LCHC

В алгоритме HCNC (High-Communication No-Extra Computation) для каждой подобласти добавляется внерасчетная граница толщиной в одну узловую точку. В начале каждого расчета происходит синхронизация между процессорами для обновления значений на внерасчетных границах. При синхронизации пограничные блоки процессора помещаются во внерасчетные границы своих соседних процессоров (рис. 3). Когда процессор в своих вычислениях подходит к границе своей подобласти и ему требуются данные с соседней подобласти - он берет эти данные со своей внерасчетной границы и продолжает вычисления.

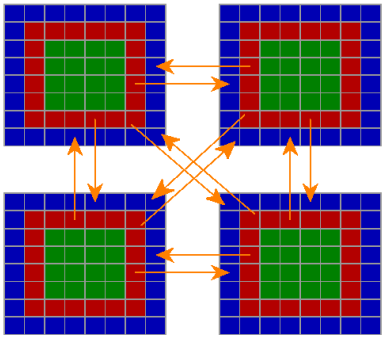

Рис. 3: Синхронизация процессоров. Красные точки - это граница подобласти, синие - внерасчетная граница

На одномерной по пространству сетке алгоритм HCNC изображен на рис. 4, слева. По вертикали отсчитываются шаги по времени, по горизонтали расположены расчетные узлы. Белым цветом обозначены расчетные узлы подобласти, красным - узлы ее внерасчетной границы, синим обведены внерасчетные узлы, которые требуют обновления своих значений на текущем временном шаге. На рисунке изображены 5 шагов алгоритма HCNC для одной из подобластей. Видно, что на каждом шаге по времени обновляется внерасчетная граница подобласти, т.е. происходит синхронизация между процессорами.

Очевидный недостаток алгоритма HCNC в том, что используется большое количество синхронизаций между процессорами, что при увеличении процессоров влечет за собой рост коммуникационных задержек и, следовательно, ухудшение масштабируемости алгоритма. Возникает идея уменьшить количество синхронизаций путем расширения внерасчетной границы области.

Этот подход хорошо иллюстрируется на одномерной по пространству сетке с толщиной внерасчетной границы в четыре узловые точки (рис. 4, справа). В начале расчета происходит синхронизация внерасчетных границ процессоров, т.е. у каждого процессора появляется по четыре значения с соседних подобластей. Затем, на шаге p происходят

вычисления уровня, адвекции и диффузии. На шаге $p+1$ происходит вычисление двумерных скоростей и переход на следующий шаг по времени. Далее опять, на шаге $p+2$ - расчет уровня, адвекции и диффузии, на шаге $p+3$ - вычисление двумерных скоростей и переход на следующий шаг по времени. Как можно заметить, во время шагов $p+1$, $p+2$, $p+3$ не происходит синхронизации между процессорами. Происходит одна большая по объему синхронизация в начале расчета и далее процессоры проводят вычисления на своей подобласти с расширенной внерасчетной границей, т.е. происходят лишние вычисления в сравнении с алгоритмом HCNC. Этот алгоритм называется LCHC (Low-Communication High-Extra Computation), т.к. он уменьшает количество синхронизаций, но при этом делает лишние вычисления.

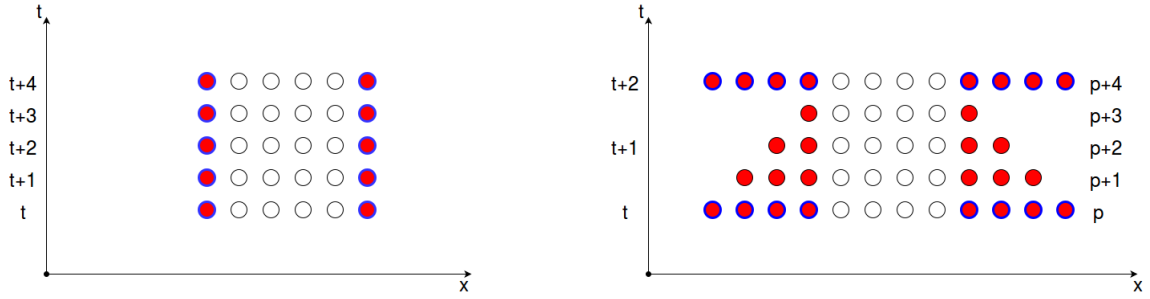

Рис. 4: Алгоритмы HCNC и LCHC на одномерной по пространству сетке

Если алгоритм HCNC требует N синхронизаций, то алгоритм LCHC с размером внерасчетной границы $bndlen$ требует $2N/bndlen$ синхронизаций. Несмотря на то, что при межпроцессорном обмене объем пересылаемой информации увеличивается в $bndlen$ раз, пересылка 1 раз $bndlen$ чисел значительно экономнее, чем $bndlen/2$ раз одно число (при условии $bndlen \geq 4$).

Такой алгоритм реализуется в параллельной версии модели общей циркуляции модели INMOM [2] для расчета быстрых баротропных гравитационных волн. Также, похожий способ параллельного расчета уравнений мелкой воды был также предложен в модели Ибраева Р.А. [11] и в работах по распараллеливанию модели циркуляции океана POM (Princeton Ocean Model) [10].

5 Балансировка нагрузки вычислений

Под балансировкой нагрузки вычислений имеется в виду такое распределение подзадач между процессорами, которое обеспечивает примерно равную нагрузку на процессоры и минимальные затраты на передачу данных между ними. Методы балансировки нагрузки вычислений позволяют снизить время простоя отдельных процессоров и повысить эффективность распараллеливания, что существенно важно при решении задач на высокопроизводительных вычислительных системах. Как уже было сказано, основная идея параллельного алгоритма решения нелинейных уравнений мелкой воды это метод декомпозиции области, и ранее было рассмотрено только равномерное разбиение на прямоугольные подобласти. Однако такое разбиение не обеспечивает балансировку нагрузки вычислений, из-за чего могут возникать сильно несбалансированные расчетные подобласти и эффективность параллельного алгоритма может ухудшиться.

Балансировка нагрузки вычислений особенно актуальна в задачах моделирования циркуляции океана, моделирования цунами. Потому что, из-за наличия берегов и островов в этих задачах равномерное разбиение будет давать особенно несбалансированные подобласти.

В работе рассматривается один из методов балансировки нагрузки: метод разбиения вдоль фрактальной кривой. Такой метод уже успел зарекомендовать себя во многих работах [12, 13].

5.1 Метод разбиения вдоль фрактальной кривой

Метод разбиения вдоль фрактальной кривой основывается на фрактальных кривых заполняющих пространство (Space-Filling Curve, SFC). Такие кривые преобразуют d-мерное пространство в одномерное с сохранением свойства локальности, т.е. соседние элементы в d-мерном пространстве преобразуются в соседние элементы в одномерном пространстве. Существует целое множество кривых заполняющих пространство [14]. На практике используются кривые Пеано и их частные случаи - кривые Гильберта и кривые Мортона. В рамках применения метода разбиения вдоль фрактальной кривой к решению нелинейных уравнений мелкой воды, мы будем рассматривать кривые Гильберта в двумерном пространстве.

Кривые Гильберта используются для преобразования двумерной области с размерами $nb_x \times nb_y$ в кривую, где $nb_x \times nb_y = 2^n$ и n - это целое число, которое называется индексом кривой Гильберта. Рис. 5 показывает пример кривых Гильберта с индексами 2, 4, 8.

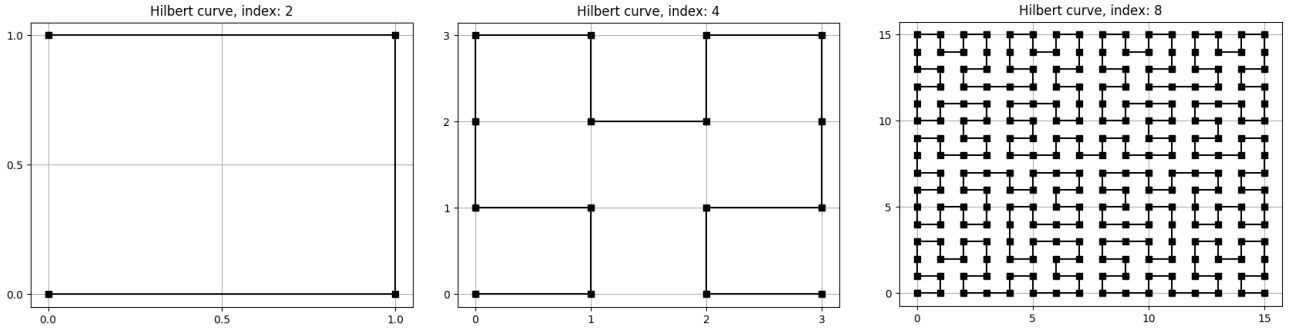

Рис. 5: Кривые Гильберта с индексами 2, 4 и 8 соответственно

Алгоритм балансировки нагрузки с помощью кривых Гильберта следующий. Предварительно вся расчетная область с размерами $nx \times ny$ точек равномерно разбивается на сетку с размерами $nb_x \times nb_y = 2^n$, состоящую из прямоугольных блоков. Здесь в качестве алгоритма разбиения берется базовый алгоритм равномерного разбиения на прямоугольные подобласти (9), только в этом случае разбиение расчетной области идет не для сетки процессоров, а для сетки блоков. При таком разбиении получается, что размер каждого блока будет $nb_1 \times nb_2$, где $nb_1 = \lceil \frac{n_x}{nb_x} \rceil$ и $nb_2 = \lceil \frac{n_y}{nb_y} \rceil$.

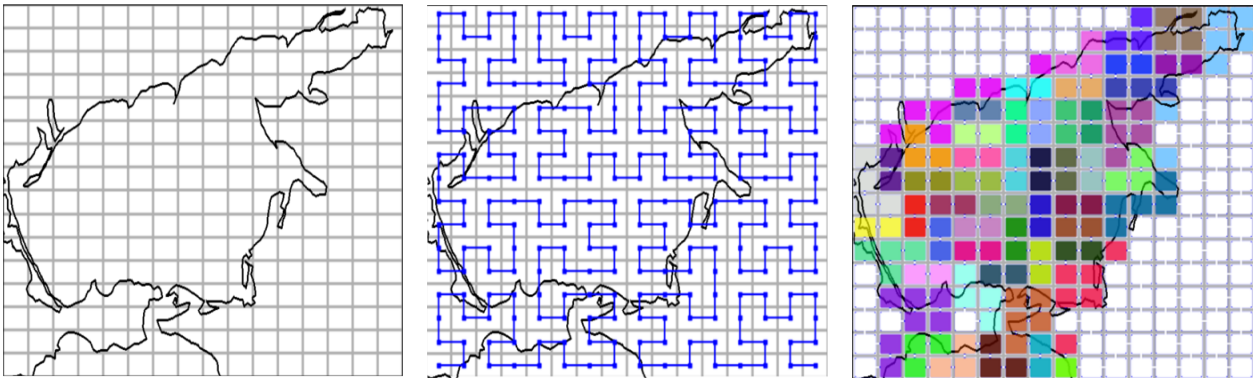

Рис. 6: Алгоритм балансировки нагрузки с помощью кривых Гильберта. 1) Равномерное разбиение на блоки 2) На сетке блоков проводится кривая Гильберта 3) Блоки объединяются в подобласти вдоль кривой Гильберта. Каждому процессору ставится в соответствие его подобласть.

Для каждого блока рассчитывается значение загруженности блока w_i, как сумма всех точек, которые не лежат на суше. Далее, на сетке блоков проводится кривая Гильберта, которая переводит двумерное пространство блоков в одномерное. Разбиение на подобласти происходит вдоль кривой и причем таким образом, чтобы подобласти имели примерно одинаковую сумму загруженности блоков. Затем каждому процессору ставится в соответствие его подобласть, на которой он проводит вычисления. Блоки, которые полностью состоят из точек на суше (т.е. загруженность которых $w_i = 0$), в распределении по процессорам и в дальнейших вычисления не участвуют. На рис. 6 наглядно показаны шаги описанного алгоритма.

Пример такого разбиения в сравнении с равномерным разбиением без балансировки нагрузки приведен на рис. 7 для акватории Азовского моря с размерами 1525×1115 точек, сетка блоков 32×32. Черным цветом на рисунке изображены блоки, состоящие только из точек на суше и которые не участвуют в вычислениях.

Введем величину LB, которая будет отвечать за несбалансированность разбиения:

$$LB = \frac{\max w_i}{\sum w_i / p} \tag{10}$$

где $\max w_i$ - максимальная загруженность блока, $\sum w_i$ - сумма загруженности всех блоков, p - количество процессоров. Эта величина показывает отношение максимальной загруженности блока в разбиении к оптимальной загруженности. Таблица показывает значения величин LB для акватории Азовского моря при разных типах разбиения, количество используемых процессоров: 64. Примеры этих разбиений показы на рис. 7 и рис. 8.

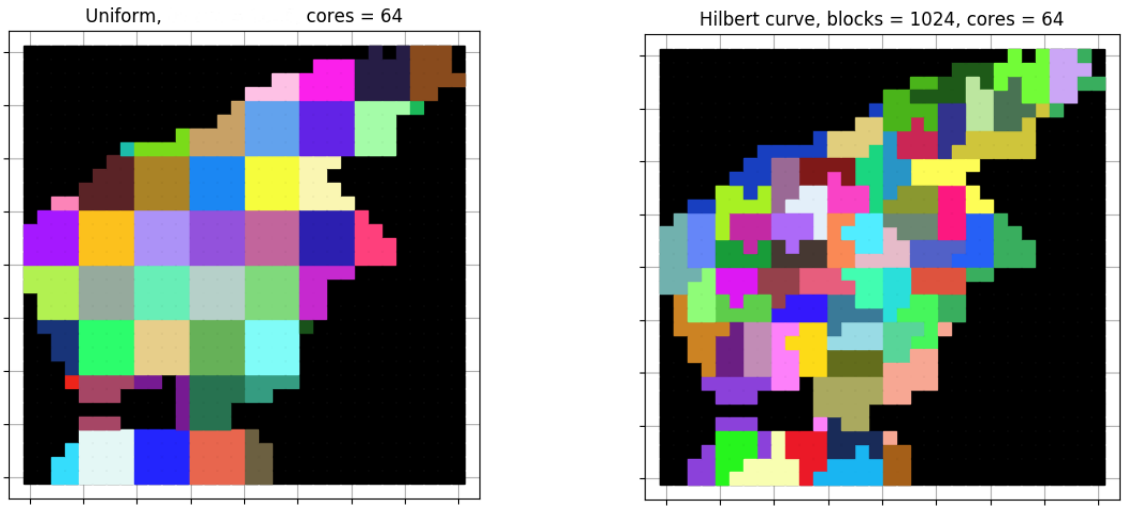

Рис. 7: Равномерное разбиение и метод разбиения с использованием кривой Гильберта для 64 процессоров

Разбиение	Равномерное	Гильберт	Гильберт	Гильберт
Количество блоков	-	256	1024	4096
Количество блоков с сушей	-	105	481	2116
LB	2.40	1.59	1.14	1.03

Видно, что с увеличением количества блоков, разбиение с использованием кривых Гильберта становится более сбалансированным. Это связано с тем, что из большого количества блоков легче формировать подобласти с примерно одинаковой загруженностью и также с тем, что чем больше блоков в разбиении, тем меньше суши они охватывают, т.к. блоки с сушей выбрасываются.

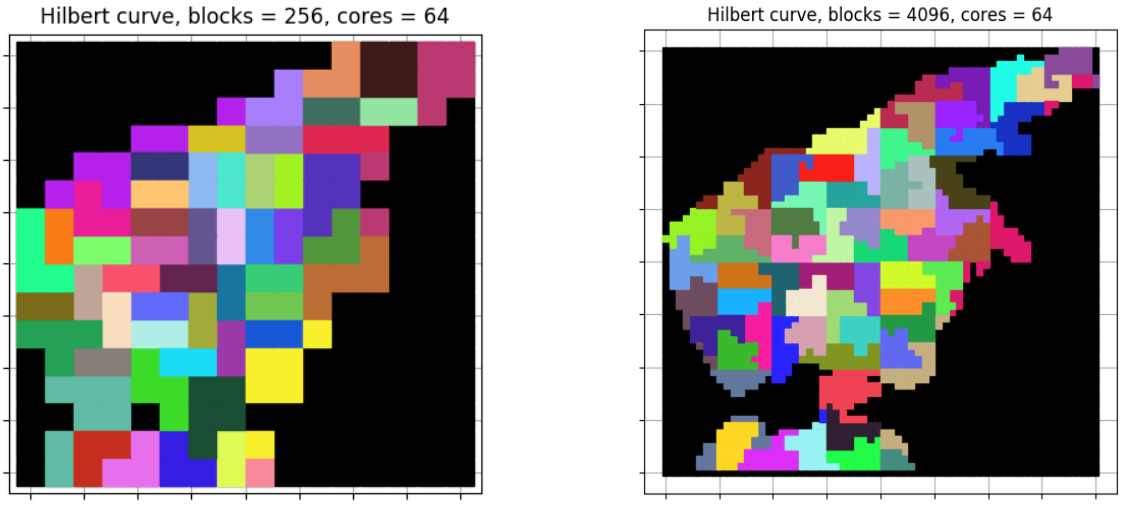

Рис. 8: Метод разбиения с использованием кривой Гильберта для 64 процессоров

Каждый блок содержит внерасчетную границу, также как в алгоритме HCNC. Коммуникации между блоками реализованы с помощью неблокирующих процедур пересылок MPI (irecv, isend). Если для блока его соседний блок находится на том же процессоре, то вызов процедуры MPI не происходит, происходит обычное копирование значений границы соседнего блока во внерасчетную границу исходного.

Работу с блоками удобно программно реализовывать, т.к. блоки представляют из себя прямоугольную область. Это позволяет нам уйти от работы на подобластях произвольной формы, чего например мы не смогли бы избежать работая с методом разбиения вдоль фрактальной кривой на сетке расчетных точек. Также выбирая небольшой размер блока, мы можем получать ускорение за счет эффективной работы с кэш памятью процессора. Похожий подход уже зарекомендовал себя в таких моделях циркуляции океана, как Parallel Ocean Program (POP) [13, 15], HIROMB [16].

Следует отметить, что у описанного метода балансировки нагрузки вычислений с использованием кривых Гильберта есть несколько недостатков: ограничение на сетку блоков (сетка должна быть размерами $nb_x \times nb_y = 2^n$); большое количество пересылок при большом количестве блоков на процессоре.

6 Масштабируемость

Тестирование параллельного алгоритма решения нелинейных уравнений мелкой воды проводилось на кластерах МВС-10П (МСЦ РАН) и ИВМ РАН.

Кластер МВС-10П состоит из 207 вычислительных узлов [17]. Каждый вычислительный узел имеет в своем составе 2 процессора Xeon E5-2690, 64 ГБ оперативной памяти, два сопроцессора Intel Xeon Phi 7110X. Коммуникационная сеть построена на базе FDR Infiniband.

Кластер ИВМ РАН состоит из головного узла, вспомогательного узла и вычислительных узлов, объеденных в разные группы [18]. В состав группы x12core входят 8 вычислительных узлов. Каждый вычислительный узел имеет два 12-ядерных процессора Intel Xeon E5-2670v3@2.30ГГц, 64 ГБ оперативной памяти. Тестирование проводилось преимущественно на группе вычислительных узлов x12core.

На обоих кластерах установлены последние версии компиляторов Intel Fortran (ifort) и библиотек MPI, с помощью которых собиралась и тестировалась программа.

На кластере МВС-10П проводилось тестирование масштабируемости алгоритмов HCNC и LCHC. Тестирование проводилось для акватории Азовского моря с разрешением 250 метров, расчетная сетка размером 1525 × 1115 точек. Размер внерасчетной границы для алгоритма LCHC 12 точек. На рис. 9 показано сравнение масштабируемости алгоритмов HCNC и LCHC, максимальное количество используемых ядер 1024. По оси y показано ускорение (speedup), вычисленное по формуле $speedup(p) = T_{init}/T_p$, где T_{init} - время работы программы на начальном количестве процессоров, T_p - время работы модели на p процессорах.

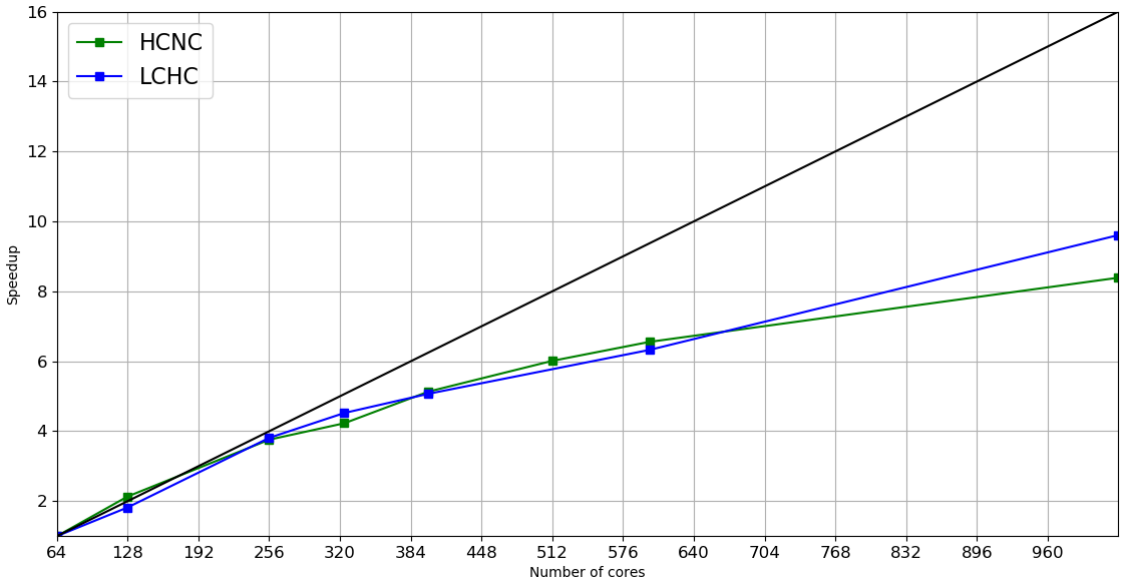

Рис. 9: Масштабируемость алгоритмов HCNC и LCHC

Из рис. 9, видно что когда число ядер сравнительно невелико, алгоритм HCNC дает даже лучше результаты чем алгоритм LCHC. Это можно объяснить тем, что время синхронизаций на небольшом количестве ядер не занимает существенно много времени и поэтому алгоритм LCHC работает хуже в силу затраты времени на лишние вычисления по сравнению с HCNC. Когда же число ядер увеличивается, время синхронизаций играет уже большую роль и алгоритм LCHC проявляет себя лучше.

На кластере ИВМ РАН проводилось тестирование масштабируемости метода балансировки нагрузки на процессоры с использованием кривых Гильберта. Тестирование проводилось также для акватории Азовского моря на сеточной области с разрешением 250 метров. На рис. 10 показана масштабируемость метода разбиения с использованием кривых Гильберта и с разными размерами блоков в сравнении с равномерным разбиением.

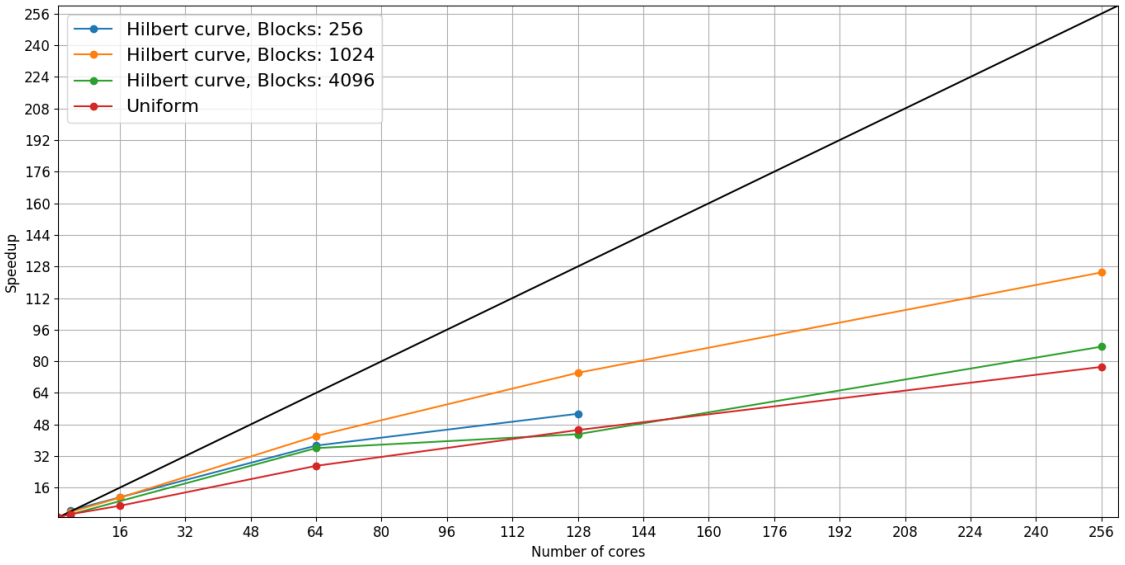

Рис. 10: Масштабируемость метода разбиения с использованием кривых Гильберта в сравнении с равномерным разбиением

Из рисунка видно, что оптимальной сеткой блоков для данной задачи является сетка с размерами 32×32. При такой сетке для данной задачи размер каждого блока составляет 1660 точек. Ухудшение масштабируемости для сетки блоков с размерами 64×64 можно объяснить тем, что при таком большом количестве блоков коммуникационные задержки на пересылки между блоками становятся существенными, хоть и подобласти при таком разбиении будут более сбалансированными (см. таблицу).

7 Моделирование цунами в Японии

С помощью параллельного алгоритма решения нелинейных уравнений мелкой воды было проведено моделирование цунами 2011 года в Японии, приведшее к катастрофе в Фукусиме, с разрешением 4 километра. Данные по начальному возвышению в очаге цунами Тохоку были любезно предоставлены М. А. Носовым [19]. Моделирование проводилось на 6 часов с шагом по времени 1 секунда, использовался коэффициент донного трения $n = 0.025$. На рис. 11 показано начальное возмущение и уровень в момент времени $t = 30$ минут.

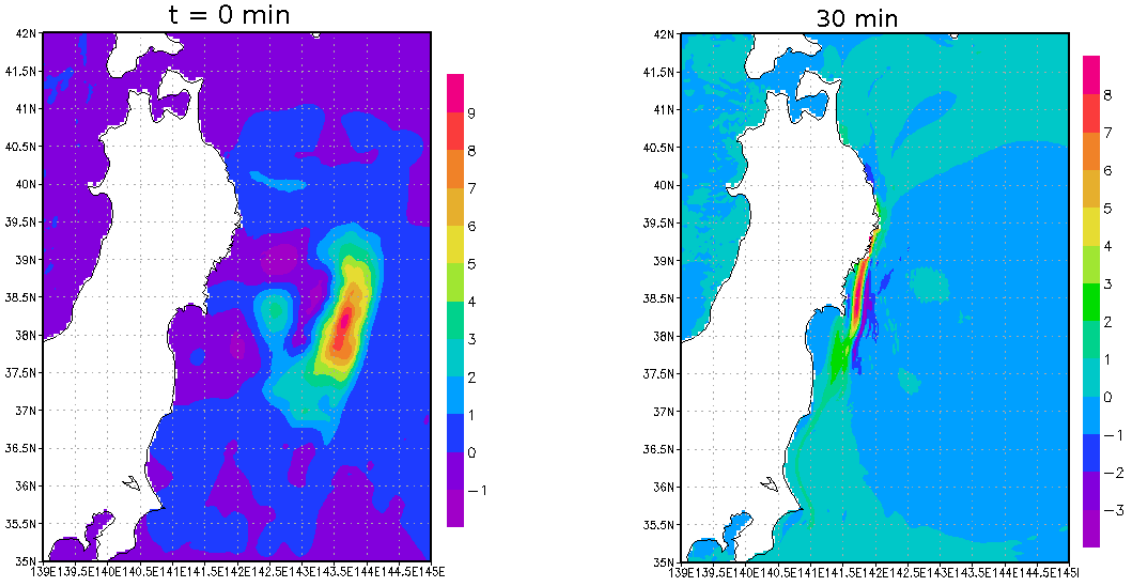

Рис. 11: Моделирование цунами в Японии, уровень (метры) в моменты времени $t = 0$ и $t = 30$ min соответственно

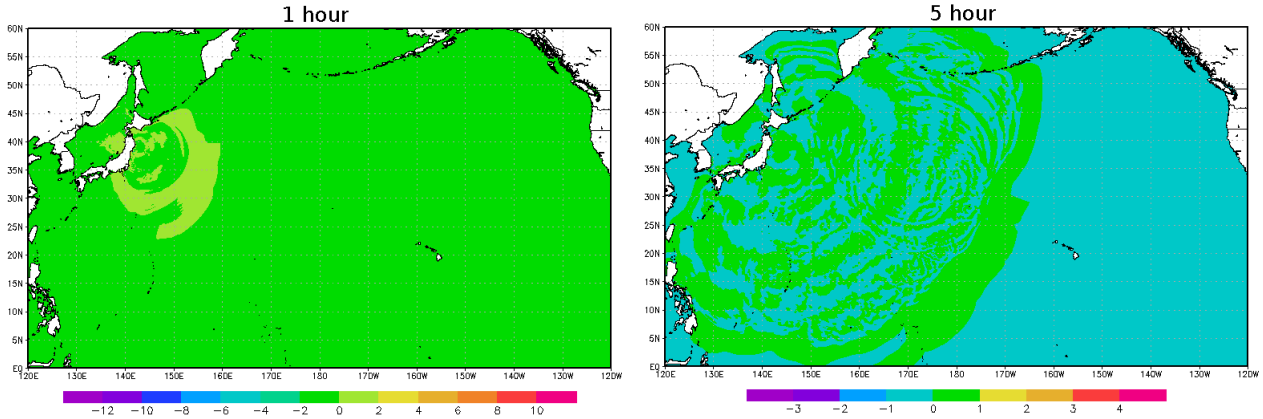

Рис. 12: Моделирование цунами в Японии, уровень (метры) в моменты времени $t = 1$ h и $t = 5$ h соответственно

Из рисунка видно, что, подходя к берегу, волна достигает 8-9 метров. Также на рис 12 показан уровень в моменты времени $t = 1$ и $t = 5$ часов. На рисунке видно как волна распространяется по акватории.

Было проведено сравнение с экспериментальными данными и сравнение результатов расчетов по нелинейным и линеаризованным уравнениям мелкой воды. Линейные уравнения мелкой воды представляют из себя более упрощенную систему чем (1):

$$\begin{aligned}
&\frac{\partial r_x r_y u}{\partial t} - r_x r_y l v + r_y g \frac{\partial \zeta}{\partial x} + r_x r_y \frac{g n^2}{h^{1/3}} u \sqrt{u^2 + v^2} = 0 \\
&\frac{\partial r_x r_y v}{\partial t} + r_x r_y l u + r_x g \frac{\partial \zeta}{\partial y} + r_x r_y \frac{g n^2}{h^{1/3}} v \sqrt{u^2 + v^2} = 0 \\
&\frac{\partial \zeta}{\partial t} + \frac{1}{r_x r_y} \left(\frac{\partial u r_y H}{\partial x} + \frac{\partial v r_x H}{\partial y} \right) = 0
\end{aligned} \tag{11}$$

В такой модели используется предположение $h \approx H$ при $\zeta << H$. Сравнение проводилось в точках, где расположены DART станции (см. таблицу).

Name	Lon	Lat
DART21418	148.694	38.711
DART21413	152.117	30.515
DART21415	171.847	50.183

На рис. 13 показано сравнение уровня, рассчитанного по линейной модели мелкой воды (11) и нелинейной (1), с данными наблюдений.

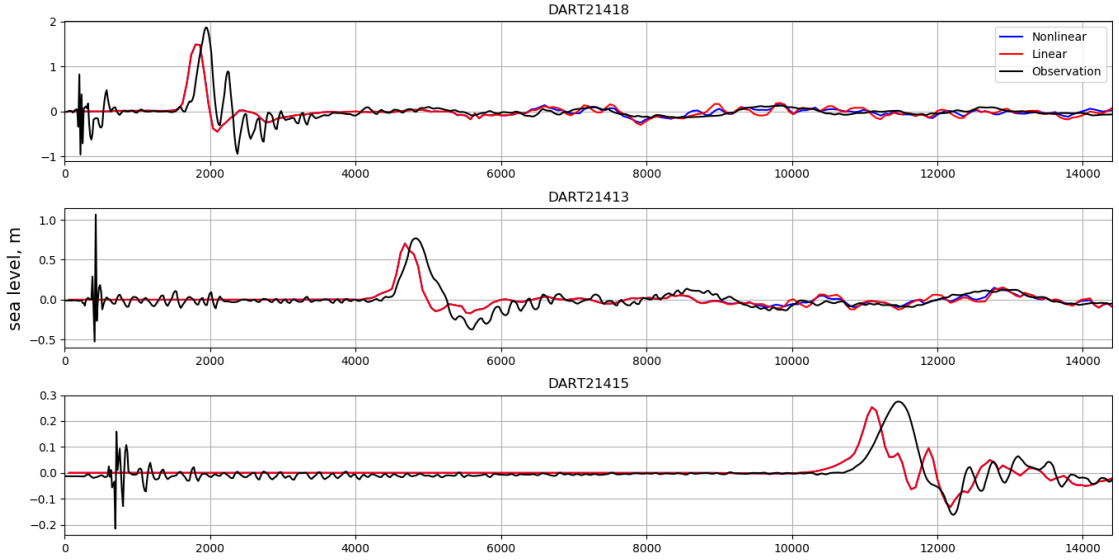

Рис. 13: Сравнение уровня (метры) в DART точках. Синий - нелинейные уравнения мелкой воды; красный - линеаризованные уравнения; черный - наблюдения

Видно, что сами модели не сильно отличаются друг друга, первый пик воспроизводят вообще идентично. Обе модели хорошо согласуются с данными реальных наблюдений. Небольшое смещение обусловлено выбранным начальным возмущением для моделей.

Точки DART расположены довольно далеко от берега, поэтому интересно посмотреть как ведут себя модели в прибрежных акваториях. На рис. 14 приведено сравнение уровня линейной и нелинейной модели для двух точек: S1, S3 (см. таблицу). Из рисунка видно, что в прибрежных акваториях разница между нелинейными уравнениями мелкой воды (1) и линеаризованными (11) может быть уже существенной. Похожий результат был получен в работе [20].

Name	Lon	Lat	H (meters)
S1	141.017	38.0167	26
S3	141.617	38.5167	108

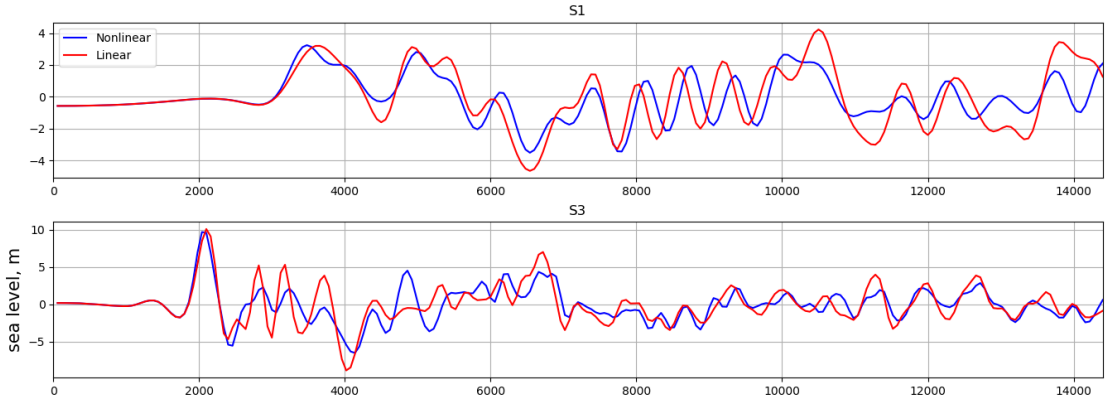

Рис. 14: Сравнение уровня (метры) в S точках. Синий - нелинейные уравнения мелкой воды; красный - линеаризованные уравнения

8 Моделирование шторма на Азовском море

Также было проведено моделирование экстремального шторма на Азовском море в марте 2013 года с разрешением 250 метров. Атмосферные данные были взяты из расчетов по WRF (Weather Research and Forecast Model) из работы [23]. Моделирование проводилось на 3 месяца с шагом по времени 1 секунда, использовался коэффициент донного трения $n = 0.025$. Было также проведено сравнение с экспериментальными данными и сравнение результатов расчетов по нелинейным (1) и линеаризованным (11) уравнениями мелкой воды (рис. 15, 16) [21]. Из рисунка видно, что обе модели хорошо согласуются с данными наблюдений.

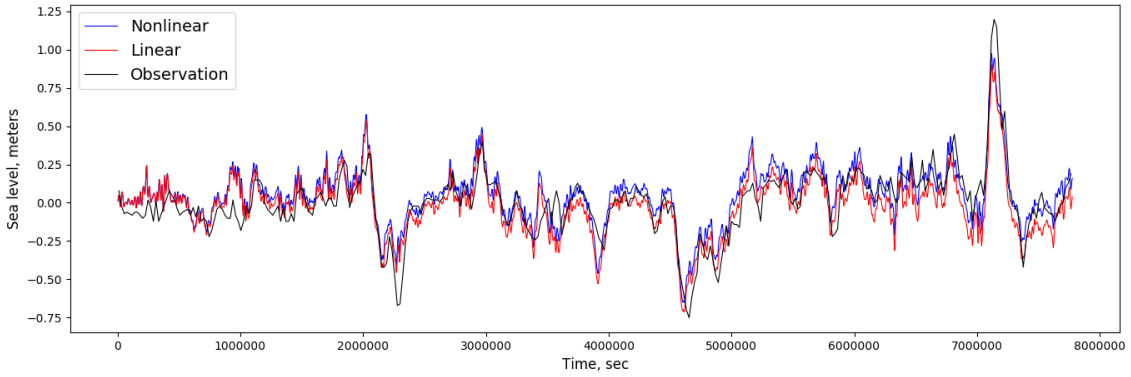

Рис. 15: Моделирование шторма в акватории Азовского моря, сравнения уровня (метры) для поста "Ейск". Синий - нелинейные уравнения мелкой воды; красный - линеаризованные уравнения; черный - наблюдения

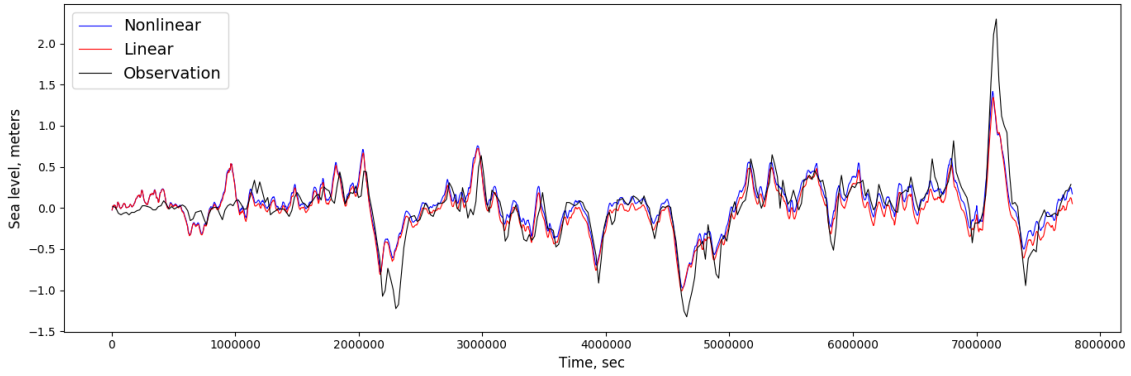

Рис. 16: Моделирование шторма в акватории Азовского моря, сравнения уровня (метры) для поста "Таганрог". Синий - нелинейные уравнения мелкой воды; красный - линеаризованные уравнения; черный - наблюдения

9 Выводы

Реализован параллельный алгоритм решения нелинейных уравнений мелкой воды в виде отдельной программы. Эту программу можно использовать как в качестве блока модели циркуляции океана INMOM, так и не зависимо, например, для расчетов прохождения волны цунами, приливов и ветрового нагона. Продемонстрирована масштабируемость двух параллельных алгоритмов решения нелинейный уравнений мелкой воды на кластере МСЦ РАН: HCNC (High-Communication No-Extra Computation), который использует большое количество пересылок между процессорами, но не использует лишние вычисления, и LCHC (Low-Communication High-Extra Computation), который уменьшает количество пересылок за счет увеличения фиктивных расчетов нерасчетной, по отношению исходному разбиению, области. Показано, что что когда число ядер сравнительно невелико, алгоритм HCNC работает эффективнее, чем алгоритм LCHC. Когда же число ядер становится более 700, увеличивается время синхронизаций и алгоритм LCHC проявляет себя лучше.

Реализована балансировка нагрузки вычислений по процессорам с использованием кривых Гильберта. Показана большая эффективность разработанного метода в сравнении с равномерным разбиением без балансировки нагрузки.

С помощью параллельной программы решения нелинейных уравнений мелкой воды было проведено воспроизведение экстремального цунами 2011 года в Японии, приведшего к катастрофе в Фукусиме, и значительного шторма на Азовском море в марте 2013. Показано, что расчет длинных волн на открытой воде практически не отличается для нелинейного и линеаризованного случаев. Однако вблизи берега расчеты значительно различаются. Очевидно, что в этом случае нелинейные уравнения более предпочтительны, т.к. более адекватно описывают здесь трансформацию длинных волн с учетом уменьшения глубин и изменений береговой черты.

В дальнейшем планируется улучшить метод балансировки нагрузки вычислений - например, попробовать использовать более сложные кривые Пеано, что позволит ослабить ограничение на размер сетки блоков. Также планируется внедрить данный метод в параллельную модель циркуляции океана INMOM для расчета уже трехмерных уравнений.

10 Список литературы

[1] Дианский Н. А. Моделирование циркуляции океана и исследование его реакции на короткопериодные и долгопериодные атмосферные воздействия. М.: Физмалит, 2012.

[2] Чаплыгин А. В. Параллельная реализация общей модели циркуляции океана INMOM. Сборник тезисов лучших выпускных квалификационных работ факультета ВМК МГУ 2017. Москва: МАКС ПРЕСС, 2017. С. 27-28.

[3] Сухов В.Б. О решении некоторых задач моделирования крупномасштабной динамики океана. Диссертация на соиск. уч. ст. к.ф.-м.н. 2009.

[4] Fumihiko Imamura, Ahmet Cedvet Yalciner, Gulizar Ozyurt. Tsunami modelling model (TUNAMI model), 2006.

[5] Роуч П. Вычислительная гидродинамика. М: Мир, 1980. 618 с.

[6] Arakawa A., Lamb V.R. Computational design of the basic dynamical processes of the UCLA general circulation model. Methods in computational Physics. V.17.

[7] Mesinger F., Arakawa A. Numerical methods used in atmospheric models. JOC, GAPR Publication Series. 1976. V.1, No. 17.

[8] Марчук Г.И. Методы вычислительной математики. М: Наука, 1989. 608 с.

[9] George L. Mellor. User guide for a three-dimensional, primitive equation, numerical ocean model. Princeton University.

[10] Guansuo Wang, Fangli Qiao, Changshui Xia. Parallelization of a coupled wave-circulation model and its application. Ocean Dynamics (2010) 60:331-339.

[11] Калмыков В. В., Ибраев Р. А. Алгоритм с перекрытиями для решения системы уравнений мелкой воды на параллельных компьютерах с распределенной памятью. Вестник УГАТУ, Т.17, No5 (58).

[12] Hui Liu, Kun Wang, Bo Yang, Min Yang, Ruijian He, Lihua Shen, He Zhong, Zhangxin Chen. Load Balancing using Hilbert Space-filling Curves for Parallel Reservoir Simulations. Cornell University Library, 4 Aug 2017.

[13] John M. Dennis. Inverse Space-Filling Curve Partitioning of a Global Ocean Model. IEEE International Parallel and Distributed Processing Symposium, 2007.

[14] Hans Sagan. Space-Filling Curves. Springer-Verlag, 1994.

[15] R. Smith, P. Jones, B. Briegleb, et al. The Parallel Ocean Program (POP) Reference Manual Ocean Component of the Community Climate System Model (CCSM) and Community Earth System Model (CESM). 23 March 2010.

[16] Tomas Wilhelmsson. Parallelization of the HIROMB Ocean Model. Licentiate Thesis, Royal Institute of Technology Department of Numerical Analysis and Computer Science, 2002.

[17] Кластер МСЦ РАН. URL: http://www.jscc.ru/scomputers.html

[18] Кластер ИВМ РАН. URL: http://cluster2.inm.ras.ru

[19] M. A. Nosov, A. V. Moshenceva, S.V. Kolesov Horizontal Motions of Water in the Vicinity of a Tsunami Source. Pure Appl. Geophys. 170 (2013), 1647-1660.

[20] Yingchun Liu, Yaolin Shi, David A. Yuen, Erik O. D. Sevre, Xiaoru Yuan, Hui Lin Xing. Comparison of linear and nonlinear shallow wave water equations applied to tsunami waves over the China Sea. Acta Geotechnica (2009) 4:129–137.

[21] Фомин В.В., Дианский Н.А., Чаплыгин А.В. Расчет экстремальных нагонов в Таганрогском заливе и использование моделей циркуляции атмосферы и океана различного пространственного разрешения. Труды VI международной научно-практической конференции «Морские исследования и образование: MARESEDU-2017», 2017.

[22] Володин Е.М., Дианский Н.А., Гусев А.В. Модель земной системы INMCM4: воспроизведение и прогноз климатических изменений в 19-21 веках. Известия РАН. Физика атмосферы и океана, 2013, т.49, №4, с 379-400.

[23] Фомин В.В., Дианский Н.А. Расчет экстремальных нагонов в Таганрогском заливе с использованием морских и атмосферных моделей различного пространственного разрешения. Метеорология и гидрология, №5, 2018.