Асинхронные параллельные алгоритмы как способ достижения 100% эффективности вычислений*

Левченко В.Д.

Институт прикладной математики им. М.В. Келдыша РАН
lev@Keldysh.ru

Аннотация

Цель работы — подбор алгоритмов, разработка методов распараллеливания и критериев их оценки, позволяющих преодолеть основные причины снижения эффективности расчетов в задачах моделирования среды при их шкалировании по размеру данных. На основе анализа тонкой информационной структуры явных сеточных методов с локальным шаблоном разработан алгоритм локальной пространственно–временной декомпозиции, использующий рекурсивное разбиение графа зависимостей, представленного в максимальной параллельной форме. Укрупнение избыточного мелкозернистого параллелизма проводится с целью локализации вычислений и создания запаса их асинхронности. Разрабатываются алгоритмы децентрализованного асинхронного управления вычислениями, основанные на методологии асинхронных клеточных автоматов. При этом используются те же структурные элементы и их связи, что и при декомпозиции графа зависимостей. Приведены результаты тестирования алгоритма, показывающие теоретическую, либо близкую к теоретической эффективность при шкалировании задачи по размеру в диапазоне 8 порядков.

1 Введение

Явные сеточные методы с локальным шаблоном (ЯЛСМ) являются классическими методами дискретизации уравнений математической физики и лежат в основе численного решения дифференциальных уравнений в частных производных гиперболического и [иногда] параболического типов. Уравнения таких типов возникают во множестве актуальных задач математической физики, включая задачи моделирования среды. Основной проблемой практического решения подобных задач является их исключительно высокая вычислительная сложность. При этом особенностью данных задач является то, что даже при использовании алгоритмов линейной сложности по данным (к которым как раз и приводят ЯЛСМ), количество обрабатываемых данных таково, что справиться с ним могут только параллельные вычислительные системы (ПВС). Из всего многообразия последних наиболее эффективными в этом смысле являются кластеры, у которых суммарная пропускная способность по доступу к данным растет линейно

*Публикуется в сборнике: Будущее вычислительной математики. М.: УРСС. 2004

при увеличении количества узлов. Исходя из этого, в работе, если не оговорено обратного, под ПВС понимаются системы с кластерной или подобной архитектурой.

Достоинством ЯЛСМ при распараллеливании является и то, что они обладают огромным запасом внутреннего параллелизма, на много порядков перекрывающим физический параллелизм, предлагаемый существующими ПВС. Казалось бы, потенциально, указанные задачи должны масштабироваться без потерь эффективности на любом доступном количестве узлов кластера. Однако, обычно из-за неоднородности загрузки узлов, в действительности этого не наблюдается. Для исправления последствий неоднородности разрабатываются различные метода балансировки загрузки [1]. Причина же проблемы заключается в несовершенстве механизмов управления и синхронизации расчетов на отдельных узлах кластера, часто базирующихся на использовании ярусно–параллельной формы (ЯПФ) представления алгоритма, что приводит к необходимости пошаговой синхронизации и, как следствие, к проблеме "бутылочного горлышка" — простоя всей ПВС до окончания выполнения самого медленного задания на каждом ярусе (обычно временном шаге) алгоритма.

Вместе с тем, пошаговая синхронизация для алгоритмов, представленных с помощью информационных графов в параллельной форме [2], не является необходимостью, и может быть заменена на управление и синхронизацию по готовности данных [3]. Для реализации этой идеи в статье привлекается аппарат асинхронных клеточных автоматов (КлА), в терминах которых переформулирован абстрактный алгоритм ЯЛСМ. Следует отметить, что клеточные автоматы являются более универсальной моделью, чем явные сеточные методы и привлекаются для описания более широкого класса явлений [4]. В этом смысле переход в терминологию КлА приводит к существенному расширению охватываемой статьей тематики. В частности, выводы статьи останутся верными также и для параллельного моделирования эволюции произвольных КлА с локальным шаблоном, а не только для порождаемых рассмотренным ЯЛСМ.

Поскольку данная работа находится на стыке нескольких областей (параллельные вычисления, физическое моделирование среды, теория клеточных автоматов), вначале следует отметить особенности используемой в статье терминологии:

Клеточный автомат (КлА) В отличие от классических клеточных автоматов с дискретным набором возможных состояний (при этом состояние можно описать вектором булевых переменных) будет использован автомат, состояние ячеек которого можно описать набором вещественных переменных (впрочем, в машинной реализации с фиксированной длиной представления чисел, второй случай можно свести к первому, хотя это неконструктивно).

Асинхронный КлА Говоря об эволюции КлА, обычно понимают эволюцию с покадровой синхронизацией как в общем модельном, так и астрономическом времени. Такие КлА в статье считаются синхронными. В отличие от них, в статье рассматриваются КлА, ячейки которых в каждый фиксированный момент астрономического времени могут относиться к разным (произвольным) кадрам. Синхронизируются только те клетки, которые связаны непосредственными зависимостями и по готовности данных. Кроме того, в отличие от асинхронных алгоритмов параллельных подстановок [5], результат эволюции КлА всегда детерминирован (в частности, эквивалентен результату эволюции синхронных КлА).

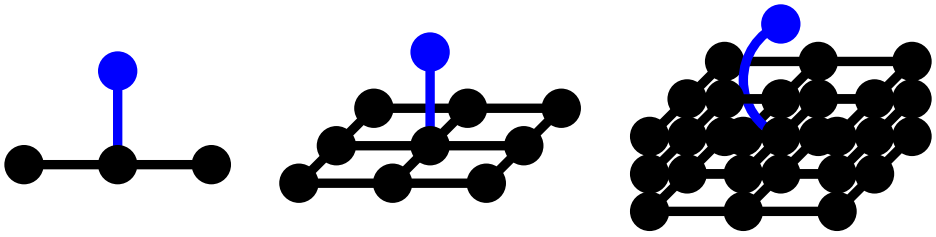

Рис. 1: Простой сеточный шаблон для одно-, двух- и трехмерной (слева направа) сеток. Черным цветом отмечены узлы сетки на текущем временном слое, синим — на следующем

2 Постановка дискретной сеточной задачи

В данной работе мы отталкиваемся от эволюционных задач моделирования среды, после дискретизации приводящих к явным локальным сеточным схемам. То есть решение на следующем временном слое явно зависит от решения на предыдущих, а сеточный шаблон для перехода с предыдущего шага на следующий ограничен в пространстве. Для определенности будем рассматривать простой четырехточечный шаблон (рис. 1 слева), а также его естественные обобщения на случай многомерной сетки с простой (расщепляемой по координатам) структурой. Часто используемые в реальных задачах шаблоны типа “крест” являются частным случаем указанных шаблонов и отдельно рассматриваться не будут, так как это не несет существенных упрощений в нашем случае. Данный шаблон соответствует шаблону Мура в теории клеточных автоматов [4]. В дальнейшем будем рассматривать только трехмерные задачи (шаблон на рис. 1 справа). Аналогичные формулы для двух и одномерных задач следуют из полученных решений тривиально.

Для определенности рассмотрим односвязную трехмерную пространственную область с простыми (в выбранной системе координат (x, y, z)) границами (например, в декартовой системе координат — параллелепипед $D(x, y, z) = [0, L_x] \times [0, L_y] \times [0, L_z]$), покрытую расщепляемой по координатам пространственной сеткой

$$G_3 = \{x_{i_x}\} \times \{y_{i_y}\} \times \{z_{i_z}\}; \quad i_x = 1 \dots N_x;\ i_y = 1 \dots N_y;\ i_z = 1 \dots N_z, \tag{1}$$

и пусть, с точностью до переобозначения координат, размеры сетки

$$N_z \geq N_y \geq N_x \gg 1. \tag{2}$$

Кроме того, существуют некоторые специальные ограничения на кратность размеров сетки степени 2, которые будут сформулированы в дальнейшем.

На сетке G_3 в начальный момент времени $t_{k=0} = 0$ в качестве начального условия зададим сеточную функцию $f^{k=0}_{i_x,i_y,i_z}$, где f — набор m вещественных чисел: $f = \{f_1, f_2, \dots, f_m\}$. Для внутренних точек области рассмотрим следующую дискретную задачу, задав для вектора решения $f^{k-1}_{i_x,i_y,i_z}$ формулу пересчета на следующий временной слой k:

$$f^{k}_{i_x,i_y,i_z} = \hat{L}(\{f^{k-1}_{i_x+d_x,i_y+d_y,i_z+d_z}\}|_{d_x,d_y,d_z \in \{-1,0,1\}}), \quad k = 1 \dots N_t$$

где $\hat{L}_{i_x,i_y,i_z}$ — произвольный оператор, зависящий от дискретного решения на текущем временном слое k в соответствие с рассмотренным трехмерным шаблоном.

Тот же оператор $\hat{L}_{i_x,i_y,i_z}$ можно использовать и для пересчета граничных ячеек, если доопределить решение для выходящих за границы сетки ячеек:

$$(f^k_{0,i_y,i_z}, f^k_{N_x+1,i_y,i_z}) = \hat{B}^{k,x}_{i_y,i_z}(f^k_{1,i_y,i_z}, f^k_{N_x,i_y,i_z}),$$
$$(f^k_{i_x,0,i_z}, f^k_{i_x,N_y+1,i_z}) = \hat{B}^{k,y}_{i_x,i_z}(f^k_{i_x,1,i_z}, f^k_{i_x,N_y,i_z}),$$
$$(f^k_{i_x,i_y,0}, f^k_{i_x,i_y,N_z+1}) = \hat{B}^{k,z}_{i_x,i_y}(f^k_{i_x,i_y,1}, f^k_{i_x,i_y,N_z}).$$

В зависимости от вида операторов $\hat{B}^{k,x}_{i_y,i_z}, \hat{B}^{k,y}_{i_x,i_z}, \hat{B}^{k,z}_{i_x,i_y}$, можно моделировать несколько типов простых граничных условий, включая граничные условия первого, второго, третьего рода, а также периодические в произвольной комбинации на всех шести граничных гранях области. Для определенности, а также в связи с некоторыми упрощениями рассуждений, в дальнейшем будем считать граничные условия трижды периодическими, то есть:

$$f^k_{i_x+N_x,i_y,i_z} = f^k_{i_x,i_y+N_y,i_z} = f^k_{i_x,i_y,i_z+N_z} = f^k_{i_x,i_y,i_z}.$$

3 Граф зависимостей эквивалентного асинхронного клеточного автомата

Построим КлА, реализующий сеточный алгоритм решения задачи по указанному шаблону. Тривиальным решением является тождественное сопоставление расчетной сетки G_3 (1) сетке ячеек КлА, значений искомой сеточной функции — вектору состояний КлА, а формулу пересчета на следующий шаг — функции перехода КлА (см. 3й столбец таблицы 1). В этом случае КлА будет обладать максимальной величиной параллельности (равной количеству ячеек сетки). Однако при организации асинхронного взаимодействия параллельно эволюционирующих ячеек возникают накладные расходы, снижающие общую производительность алгоритма. Синхронная же покадровая эволюция КлА, хоть и лишена накладных расходов, по сути будет полностью аналогична ярусно–параллельной форме представления исходного сеточного алгоритма с вытекающими отсюда ограничениями при масштабируемости (известная проблема бутылочного горлышка, когда скорость расчета в параллельной системе определяется самым медленным ее компонентом).

Вместе с тем, уровень мелкозернистого параллелизма, содержащегося в исходной дискретной задаче (количество узлов сетки) для эффективной реализации на современных компьютерах всегда избыточен [6] (на 5-7 и более десятичных порядков), кроме того, указанный разрыв имеет тенденцию к увеличению с развитием технологий производства RAM. Следовательно, без какого-либо ущерба, избыточным параллелизмом можно пожертвовать, проведя композицию элементарных параллельных операций, перейдя таким образом к среднезернистому параллелизму.

В данной работе подобная композиция проводится в два этапа. Первый связан с редукцией (уменьшением) пространственной размерности сетки КлА и приводит к снижению доли накладных расходов на организацию асинхронного взаимодействия до пренебрежимо малого уровня, одновременно снижая количество межклеточных связей. На этом уровне также могут быть эффективно задействованы векторные или матричные вычислители (при их наличии). Второй (описан в разделе 5) связан с рекурсивной декомпозицией графа зависимостей

параметр	1D	2D	3D
Сетка	G_1	G_2	G_3
Вектор состояния	$f^{C,k}_{i_z} = \left\{ f^k_{i_x,i_y,i_z} \right\}$, $i_x = 1, \dots N_x$; $i_y = 1, \dots N_y$	$f^{C,k}_{i_y,i_z} = \left\{ f^k_{i_x,i_y,i_z} \right\}$, $i_x = 1, \dots N_x$	$f^{C,k}_{i_x,i_y,i_z} = f^k_{i_x,i_y,i_z}$
Функция перехода	$\mathcal{C}\left(\{ f^{C,k-1}_{i_x,i_y,i_z+d_z} \}\vert_{d_z \in \{-1,0,1\}} \right)$	$\mathcal{C}\left(\{ f^{C,k-1}_{i_x,i_y+d_y,i_z+d_z} \}\vert_{d_y,d_z \in \{-1,0,1\}} \right)$	$\mathcal{C}\left(\{ f^{C,k-1}_{i_x+d_x,i_y+d_y,i_z+d_z} \}\vert_{d_x,d_y,d_z \in \{-1,0,1\}} \right)$
Параллелизм	N_z	$N_y N_z$	$N_x N_y N_z$
Узлов в шаблоне	3	9	27
расчеты/проверки	$(N_x N_y)/2$	$N_x/8$	$1/26$
расчеты/пересылки	$N_z/4N_P$	$\sqrt{N_y N_z/8N_P}$	$\sqrt[3]{N_x N_y N_z/12N_P}$

Таблица 1: Определение КлА для сеток различной размерности и сравнение параметров их эффективности. Пересылки учтены в модели пространственной декомпозиции области на N_P равносторонних частей

алгоритма с целью повышения уровня асинхронности, и одновременно приводит к автоматической пространственно–временной локализации данных, что позволяет эффективно использовать всю иерархию памяти вычислительных узлов, включая кэши процессоров, оперативную и дисковую память.

Редукция пространственной размерности сетки КлА. Для контроля за накладными расходами при эволюции КлА, вносимыми его реализацией, вначале дополнительно к уже заданной 3D разностной сетке G_3 (1) введем также редуцированные сетки меньшей размерности (2D: $G_2 = \{y_{i_y}\} \times \{z_{i_z}\}$ и 1D: $G_1 = \{z_{i_z}\}$). В зависимости от того, какая из сеток будет использована для дальнейших построений КлА, будем говорить о 3D, 2D или 1D сетке КлА, и, на втором этапе, о 3D, 2D или 1D локальной декомпозиции соответственно. Следует отметить, что в любом случае исходная задача остается трехмерной по пространству, при этом каждая ячейка КлА будет отвечать за эволюцию 1, N_x или $N_x \times N_y$ ячеек пространственной сетки дискретной задачи. На выбранной сетке зададим КлА `Cell` в соответствии с таблицей 1.

Учитывая соглашение о размерах сетки (2), а также опыт решения подобных задач на кластерах, последние две строки таблицы указывают на то, что 2D сетка КлА будет наиболее эффективным решением для трехмерных задач. Однако, в целях универсальности, мы продолжим расмотрение 2D совместно с 1D и 3D вариантами. Кроме того, предложенный в разделе 5 алгоритм позволяет вообще отказаться от проверок состояния отдельных ячеек КлА.

Асинхронные (асинхронно управляемые) клеточные автоматы. В теории КлА при переходе от предыдущего кадра k к следующему кадру $k+1$ векторы состояний f^C всех ячеек обновляются параллельно. Прямое воплощение подобного подхода возможно лишь на вычислительных системах, имеющих уровень параллельности не меньший чем само количество КлА. Но для рассматриваемых задач это просто неактуально, даже для редуцированной 1D сетки КлА.

Так как алгоритм пересчета состояния КлА с кадра на кадр соответствует ярусно–параллельной форме (ЯПФ), то, следуя методологии ЯПФ, в реальных алгоритмах параллельность переходов ячеек КлА часто заменяют покадровой синхронизацией. Иначе говоря, вычисление вектора состояний f_i^{k+1} начинают после вычисления всех значений f_i^k. Такой же подход — пошаговое выполнение с синхронизацией по окончании каждого временного шага — будучи очевидным, повсеместно применяется и при решении эволюционных задач математической физики. Тем не менее он обладает рядом очевидных недостатков.

Первый (потенциально преодолимый) недостаток состоит в том, что при выполнении операции перехода необходимо резервировать память для промежуточного хранения вычисляемого вектора состояния КлА. Второй, более серьезный недостаток, связан с самой сутью покадровой синхронизации и приводит к снижению эффективности распараллеливания при масштабировании, и связан с тем,что время перехода КлА с кадра на кадр определяется временем выполнения самого медленного из параллельных заданий. Кроме того, накладные расходы коммуникаций при синхронизации в этом случае не маскируются. Таким образом возникают неустранимые накладные расходы, приводящие к деградации эффективности распараллеливания с ростом числа узлов ПВС.

Для устранения причины указанной проблемы следует заменить покадровую синхронизацию КлА асинхронным децентрализованным управлением вычислениями, используя методологию асинхронных КлА. Для введения в рассмотрение асинхронных КлА понятие текущего кадра КлА k достаточно заменить текущим состоянием ячейки КлА: для ячейки i текущее состояние $F_i = k$, если $f_i^{C,k}$ уже вычислено, а $f_i^{C,k+1}$ — еще нет. Кроме того, дополнительно к правилам перехода, необходимо определить условия выполнения перехода (предикаты): переход $\mathcal{C}\left(f_i^{k_i} | i \in \mathtt{St}\right)$ допустим тогда и только тогда, когда текущие состояния каждого из узлов шаблона `St` достигло указанного в функции перехода $\mathcal{C}$ номера кадра k_i: $F_i \geq k_i, \forall i \in \mathtt{St}$.

Что же касается проблемы выделения дополнительной памяти для промежуточного вектора состояний, то в случае асинхронных КлА ситуация усугубляется неоднозначностью порядка выполнения переходов для ячеек КлА, что приводит в общем случае к необходимости уже постоянного хранения состояния ячейки с предыдущего кадра.

В данной работе для хранения промежуточной информации (относящейся как к вектору состояний, так и к асинхронному управлению) избран другой подход: дополнительно к уже определенному КлА `Cell` в рассмотрение вводятся вспомогательные автоматы, заданные на сдвинутых сетках. Полученная при этом система взаимодействующих автоматов будет эквивалентна исходному КлА. Обоснование данного выбора сводится к двум фактам:

- С одной стороны, часто используемый при дискретизации уравнений в частных производных интегро–интерполяционный метод [7] и его аналоги приводят к появлению промежуточных сеточных значений, определенных на сдвинутых сетках и в полуцелые моменты времени, то есть для рассмотренного случая дополнительной памяти может и не потребоваться (как, например, в рассмотренном далее примере).

- С другой стороны, рассмотренная в данной работе реализация асинхронного управления при среднезернистом параллелизме имеет структуру связей, подобную структуре связей, возникающих на сдвинутых сетках. Таким образом, в задаче возникает рекурсивное структурное подобие связей данных и управления, пронизывающее ее от самого мелкого масштаба до масштаба, соответствующего среднезернистому параллелизми, и далее вплоть до масштаба размеров всей системы.

обозначение автомата				точка привязки на сетке 1D	2D	3D	описание (точка привязки)
Cell	f^C	$\mathtt{St}^{\mathtt{C}}$	C	(z_{i_z})	(y_{i_y}, z_{i_z})	$(x_{i_x}, y_{i_y}, z_{i_z})$	ячейка сетки (центр ячейки)
Mesh	f^M	$\mathtt{St}^{\mathtt{M}}$	M	$(z_{i_z \pm \frac{1}{2}})$	$(y_{i_y \pm \frac{1}{2}}, z_{i_z \pm \frac{1}{2}})$	$(x_{i_x \pm \frac{1}{2}}, y_{i_y \pm \frac{1}{2}}, z_{i_z \pm \frac{1}{2}})$	узел сетки
Side	f_z^S	$\mathtt{St}_{\mathtt{z}}^{\mathtt{S}}$	$\mathtt{S}_{\mathtt{z}}$	нет	$(y_{i_y}, z_{i_z \pm \frac{1}{2}})$	$(x_{i_x}, y_{i_y}, z_{i_z \pm \frac{1}{2}})$	грань, разделяющая ячейки (центр линии, соединяющей центры соседних ячеек)
	f_y^S	$\mathtt{St}_{\mathtt{y}}^{\mathtt{S}}$	$\mathtt{S}_{\mathtt{y}}$	нет	$(y_{i_y \pm \frac{1}{2}}, z_{i_z})$	$(x_{i_x}, y_{i_y \pm \frac{1}{2}}, z_{i_z})$	
	f_x^S	$\mathtt{St}_{\mathtt{x}}^{\mathtt{S}}$	$\mathtt{S}_{\mathtt{x}}$	нет	нет	$(x_{i_x}, y_{i_y}, z_{i_z \pm \frac{1}{2}})$	
Edge	f_z^E	$\mathtt{St}_{\mathtt{z}}^{\mathtt{E}}$	$\mathtt{E}_{\mathtt{z}}$	нет	нет	$(x_{i_x \pm \frac{1}{2}}, y_{i_y \pm \frac{1}{2}}, z_{i_z})$	пересечение разделяющих ячейки линий (центр линии, соединяющей соседние узлы)
	f_y^E	$\mathtt{St}_{\mathtt{y}}^{\mathtt{E}}$	$\mathtt{E}_{\mathtt{y}}$	нет	нет	$(x_{i_x \pm \frac{1}{2}}, y_{i_y}, z_{i_z \pm \frac{1}{2}})$	
	f_x^E	$\mathtt{St}_{\mathtt{x}}^{\mathtt{E}}$	$\mathtt{E}_{\mathtt{x}}$	нет	нет	$(x_{i_x}, y_{i_y \pm \frac{1}{2}}, z_{i_z \pm \frac{1}{2}})$	

Таблица 2: Типы вспомогательных КлА и их привязка на сетке. В первых четырех столбцах указаны обозначения самого автомата, вектора его состояния, шаблона и функции перехода соответственно

Автоматы на сдвинутых сетках. Точки пространства $(x_{i_x}, y_{i_y}, z_{i_z})$, будем считать центрами ячеек сетки, также для удобства введем промежуточные сеточные значения (узлы):

$$x_{i+\frac{1}{2}} = x_i + \alpha_i^x (x_{i+1} - x_i), \quad y_{i+\frac{1}{2}} = y_i + \alpha_i^y (y_{i+1} - y_i), \quad z_{i+\frac{1}{2}} = z_i + \alpha_i^z (z_{i+1} - z_i),$$

где $0 < \alpha^x, \alpha^y, \alpha^z < 1$, а для равномерной сетки $\alpha^x = \alpha^y = \alpha^z = 1/2$. Введем построенную на узлах сдвинутую сетку

$$G_3^M = \left\{x_{i_x+\frac{1}{2}}\right\} \times \left\{y_{i_y+\frac{1}{2}}\right\} \times \left\{z_{i_z+\frac{1}{2}}\right\}; \quad i_x = 0 \dots N_x;\ i_y = 0 \dots N_y;\ i_z = 0 \dots N_z, \tag{3}$$

охватывающую сетку центров ячеек (1). На сетке КлА выбранной размерности введем клеточные автоматы нескольких типов (таблица 2). Остается определить шаблоны и правила перехода для полученной системы связанных КлА со слоя $k-1$ на слой k в случаях различных размерностей.

1D КлА:

$$\mathtt{St}^{\mathtt{C}} = \{i_z \pm \tfrac{1}{2}\}, \qquad f_{i_z}^{C,k} = \mathtt{C}\left(f_{i_z}^{C,k-1}, f_i^{M,k} | i \in \mathtt{St}^{\mathtt{C}}\right);$$
$$\mathtt{St}^{\mathtt{M}} = \{i_z - 1, i_z\}, \qquad f_{i_z-\frac{1}{2}}^{M,k+1} = \mathtt{M}\left(f_{i_z-\frac{1}{2}}^{M,k}, f_i^{C,k} | i \in \mathtt{St}^{\mathtt{M}}\right).$$

2D КлА:

$$\begin{aligned}
\mathtt{St}^{\mathtt{C}} &= \{(i_y, i_z \pm \tfrac{1}{2}), (i_y \pm \tfrac{1}{2}, i_z)\}, & f^{C,k}_{i_y,i_z} &= \mathtt{C}\left(f^{C,k-1}_{i_y,i_z}, f^{S,k-1}_i | i \in \mathtt{St}^{\mathtt{C}}\right);\\
\mathtt{St}^{\mathtt{S},\mathtt{1}}_{\mathtt{z}} &= \{(i_y, i_z), (i_y, i_z - 1)\}, & f^{S1,k+1}_{i_y,i_z-\frac{1}{2}} &= \mathtt{S}^{\mathtt{1}}_{\mathtt{z}}\left(f^{S,k}_{i_y,i_z-\frac{1}{2}}, f^{C,k+1}_i | i \in \mathtt{St}^{\mathtt{S},\mathtt{1}}_{\mathtt{z}}\right);\\
\mathtt{St}^{\mathtt{S},\mathtt{1}}_{\mathtt{y}} &= \{(i_y, i_z), (i_y - 1, i_z)\}, & f^{S1,k+1}_{i_y-\frac{1}{2},i_z} &= \mathtt{S}^{\mathtt{1}}_{\mathtt{y}}\left(f^{S,k}_{i_y-\frac{1}{2},i_z}, f^{C,k+1}_i | i \in \mathtt{St}^{\mathtt{S},\mathtt{1}}_{\mathtt{y}}\right);\\
\mathtt{St}^{\mathtt{M}} &= \left\{(i_y - d_y, i_z - d_z)|_{d_y,d_z \in \{0,1\}}\right\}, & f^{M,k+1}_{i_y-\frac{1}{2},i_z-\frac{1}{2}} &= \mathtt{M}\left(f^{M,k}_{i_y-\frac{1}{2},i_z-\frac{1}{2}}, f^{S1,k+1}_i | i \in \mathtt{St}^{\mathtt{M}}\right);\\
\mathtt{St}^{\mathtt{S},\mathtt{2}}_{\mathtt{z}} &= \{(i_y \pm \tfrac{1}{2}, i_z - \tfrac{1}{2})\}, & f^{S,k+1}_{i_y,i_z-\frac{1}{2}} &= \mathtt{S}^{\mathtt{2}}_{\mathtt{z}}\left(f^{S1,k+1}_{i_y,i_z-\frac{1}{2}}, f^{M,k+1}_i | i \in \mathtt{St}^{\mathtt{S},\mathtt{2}}_{\mathtt{z}}\right);\\
\mathtt{St}^{\mathtt{S},\mathtt{2}}_{\mathtt{y}} &= \{(i_y - \tfrac{1}{2}, i_z \pm \tfrac{1}{2})\}, & f^{S,k+1}_{i_y-\frac{1}{2},i_z} &= \mathtt{S}^{\mathtt{2}}_{\mathtt{y}}\left(f^{S1,k+1}_{i_y-\frac{1}{2},i_z}, f^{M,k+1}_i | i \in \mathtt{St}^{\mathtt{S},\mathtt{2}}_{\mathtt{y}}\right).
\end{aligned}$$

3D КлА (с целью более компактной записи для КлА `Side` и `Edge` приведены выражения только для одной компоненты из трех (именно, x), формулы для y и z компонент можно получить циклической перестановкой координат, отметка $|_{\text{cicle}}$ также означает перебор циклической перестановкой).

$$\begin{aligned}
\mathtt{St}^{\mathtt{C}} &= \{(i_x \pm \tfrac{1}{2}, i_y, i_z)\}|_{\text{cicle}}, & f^{C,k}_{i_x,i_y,i_z} &= \mathtt{C}\left(f^{C,k-1}_{i_x,i_y,i_z}, f^{S,k-1}_i | i \in \mathtt{St}^{\mathtt{C}}\right);\\
\mathtt{St}^{\mathtt{S},\mathtt{1}}_{\mathtt{x}} &= \{(i_x, i_y, i_z), (i_x - 1, i_y, i_z)\}, & f^{S1,k+1}_{i_x-\frac{1}{2},i_y,i_z} &= \mathtt{S}^{\mathtt{1}}_{\mathtt{x}}\left(f^{S,k}_{i_x-\frac{1}{2},i_y,i_z}, f^{C,k+1}_i | i \in \mathtt{St}^{\mathtt{S},\mathtt{1}}_{\mathtt{x}}\right);\\
\mathtt{St}^{\mathtt{E},\mathtt{1}}_{\mathtt{x}} &= \left\{(i_x, i_y - d_y, i_z - d_z)|_{d_y,d_z \in \{0,1\}}\right\}, & f^{E1,k+1}_{i_x,i_y-\frac{1}{2},i_z-\frac{1}{2}} &= \mathtt{E}^{\mathtt{1}}_{\mathtt{x}}\left(f^{E,k}_{i_x,i_y-\frac{1}{2},i_z-\frac{1}{2}}, f^{S1,k+1}_i | i \in \mathtt{St}^{\mathtt{E},\mathtt{1}}_{\mathtt{x}}\right);\\
\mathtt{St}^{\mathtt{M}} &= \left\{(i_x - d_x, i_y - \tfrac{1}{2}, i_z - \tfrac{1}{2})|_{d_x \in \{0,1\}}\right\}|_{\text{cicle}}, & f^{M,k+1}_{i_x-\frac{1}{2},i_y-\frac{1}{2},i_z-\frac{1}{2}} &= \mathtt{M}\left(f^{M,k}_{i_x-\frac{1}{2},i_y-\frac{1}{2},i_z-\frac{1}{2}}, f^{E1,k+1}_i | i \in \mathtt{St}^{\mathtt{M}}\right);\\
\mathtt{St}^{\mathtt{E},\mathtt{2}}_{\mathtt{x}} &= \{(i_x \pm \tfrac{1}{2}, i_y - \tfrac{1}{2}, i_z - \tfrac{1}{2})\}, & f^{E,k+1}_{i_x,i_y-\frac{1}{2},i_z-\frac{1}{2}} &= \mathtt{E}^{\mathtt{2}}_{\mathtt{x}}\left(f^{E1,k+1}_{i_x,i_y-\frac{1}{2},i_z-\frac{1}{2}}, f^{M,k+1}_i | i \in \mathtt{St}^{\mathtt{E},\mathtt{2}}_{\mathtt{x}}\right);\\
\mathtt{St}^{\mathtt{S},\mathtt{2}}_{\mathtt{x}} &= \{(i_x - \tfrac{1}{2}, i_y \pm \tfrac{1}{2}, i_z \pm \tfrac{1}{2})\}, & f^{S,k+1}_{i_x-\frac{1}{2},i_y,i_z} &= \mathtt{S}^{\mathtt{2}}_{\mathtt{x}}\left(f^{S1,k+1}_{i_x-\frac{1}{2},i_y,i_z}, f^{E,k+1}_i | i \in \mathtt{St}^{\mathtt{S},\mathtt{2}}_{\mathtt{x}}\right);
\end{aligned}$$

Возвращаясь теперь к вопросу об эквивалентности построенной системы связанных автоматов исходному КлА на простой сетке, проведем прямую последовательную подстановку:

$$\mathtt{C}(f^C, \mathtt{S}^{\mathtt{2}}(\mathtt{S}^{\mathtt{1}}(f^S, f^C), \mathtt{E}^{\mathtt{2}}(\mathtt{E}^{\mathtt{1}}(f^E, \mathtt{S}^{\mathtt{1}}(f^S, f^C)), \mathtt{M}(f^M, \mathtt{E}^{\mathtt{1}}(f^E, \mathtt{S}^{\mathtt{1}}(f^S, f^C))))))),$$

которая связывает результирующее правило перехода совокупного вектора состояний $f = \{f^C, f^E, f^S, f^M\}_{i_x,i_y,i_z}$ с аналогом исходного КлА: $f^{k+1}_{i_x,i_y,i_z} = \mathcal{C}(f^k_{i_x,i_y,i_z}, f^k_i)|_{i \in \mathtt{St}}$, при этом совпадает и шаблон: $\mathtt{St} = \{(i_x \pm 1, i_y \pm 1, i_z \pm 1)\}$.

4 Пространственно–временная декомпозиция информационного графа алгоритма

Информационные зависимости между КлА на сдвинутых сетках представлены на рис. 2. Для одномерной сетки (рис. 2, слева) представлен фрагмент полного графа зависимостей в параллельной форме, для двух- и трех- мерных сеток (рис. 2, в центре и справа) — проекции

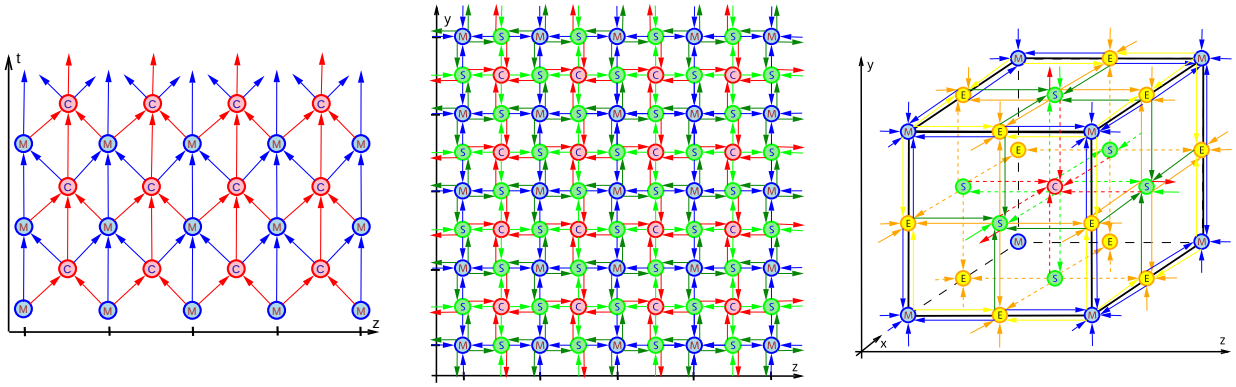

Рис. 2: Зависимости по данным системы связанных КлА для одно-, двух- и трехмерной (слева направа) сеток. Вспомогательные КлА: `Cell` (красный), `Mesh` (синий), `Side` (зеленый), `Edge` (желтый)

этого графа на пространственные координаты (или, по-другому, показаны соседние ярусы параллельной формы, при этом остальные ярусы идентичны показанным и повторяются периодически). Под графом зависимостей здесь понимается направленный ациклический граф алгоритма, построенный в соответствии со следующими правилами:

- В вершины (узлы) графа зависимостей помещены элементарные операции алгоритма (ими в случае КлА являются функции перехода для каждой ячейки).
- Входящим в каждую вершину ребрам приписаны входные данные для каждой операции (соответствуют векторам состояния ячеек шаблона КлА),
- Выходящим из вершины графа ребрам соответствует результат операции (вектор состояния ячейки КлА или его часть).

Параллельная форма графа означает, что проведено упорядочение вершин по уровню их зависимостей, то есть, узлы графа (операции) расположены на перенумерованных ярусах (на рис. 2, слева, нумерация снизу вверх по оси t), причем операция на ярусе k_1 зависит от результатов только тех операций, которые расположены на ярусах с меньшими номерами ($k_i < k$). При этом все операции, находящиеся на каждом ярусе параллельной формы графа зависимостей, являются независимыми и могут выполняться параллельно. Таким образом, уровень внутреннего параллелизма измеряется средним количеством операций на ярусе и имеет место так называемый мелкозернистый параллелизм.

Из рисунка ясно, что в рассматриваемом случае уровень внутреннего параллелизма равен N_z, N_zN_y и $N_zN_yN_x$ для сеток размерности 1D, 2D и 3D соответственно, кроме того, учитывая принятое соглашение о размерах сетки (2), уровень внутреннего параллелизма во всех случаях заведомо избыточен для обычных ПВС, количество параллельных процессов в которых ограничено сотнями (до нескольких тысяч) штук.

В связи с этим, при распараллеливании, встает вопрос об эффективном отражении мелкозернистого параллелизма задачи на ограниченное число параллельных процессов ПВС [8],

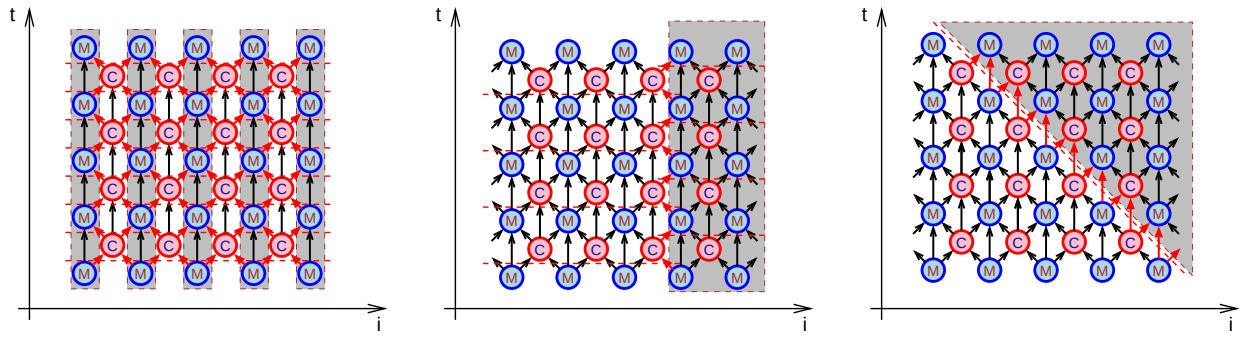

Рис. 3: Фрагмент декомпозиции 1D графа зависимостей для максимально-параллельной формы с поярусной синхронизацией (ЯПФ), метода SDD и обсуждаемого в статье метода LSTD (слева направо). Белым и серым фоном выделены разные процессы, черными стрелками — внутренние связи, красными — граничные, красным пунктиром отмечены моменты синхронизации процессов

Метод	ЯПФ	SDD			LSTD		
		1D	2D	3D	1D	2D	3D
Доля обменов	1	$\frac{N_P}{N_z}$	$\sqrt{\frac{N_P}{N_y N_z}}$	$\sqrt[3]{\frac{N_P}{N_x N_y N_z}}$	$\frac{N_P}{N_z}$	$\sqrt{\frac{N_P}{N_y N_z}}$	$\sqrt[3]{\frac{N_P}{N_x N_y N_z}}$
Запас асинхронности процесса	$\frac{1}{N_T}$	$\frac{1}{N_T}$	$\frac{1}{N_T}$	$\frac{1}{N_T}$	$\frac{N_z}{N_P N_T}$	$\frac{\sqrt{N_y N_z / N_P}}{N_T}$	$\frac{\sqrt[3]{N_x N_y N_z / N_P}}{N_T}$
задачи	$\frac{1}{N_T}$	$\frac{N_P}{N_T}$	$\frac{\sqrt{N_P}}{N_T}$	$\frac{\sqrt[3]{N_P}}{N_T}$	$\frac{N_z}{N_T}$	$\frac{N_y}{N_T}$	$\frac{N_x}{N_T}$

Таблица 3: Параметры эффективности декомпозиции графа зависимостей (указан порядок)

иными словами, о распределении множества элементарных операций алгоритма между параллельными процессами с учетом информационных связей между этими операциями. В терминах графа зависимостей, можно говорить о его декомпозиции — разбиении на подграфы, каждый из которых связывается с конкретным процессом ПВС.

Иллюстрации трех различных способов декомпозиции графа зависимостей для случая одномерной сетки приведены на рис. 3.

В качестве критерия эффективности того или иного способа декомпозиции обычно выдвигается минимизация объема пересылок данных между параллельными процессами, что в терминах графа зависимостей означает минимизацию отношения количества граничных связей, L^{BC} (ребер исходного графа, связывающих вершину, принадлежащую подграфу процесса с вершиной, ему не принадлежащей) к количеству внутренних связей, L^{in} (тех ребер, которые связывают два принадлежащих подграфу процесса вершины исходного графа). Возможно, с учетом весов каждой связи (если они неравнозначны). Из рис. 3 видно, что для ЯПФ уровень обменов $L^{\text{BC}}/L^{\text{in}} = O(1)$, в то время как метод декомпозиции пространственной области (SDD) [9] и обсуждаемый в статье метод LSTD [10, 11] имеют один и тот же порядок уровня обменов $O(N_P/N_z)$ (здесь, N_P — число параллельных процессов). Доля обменов для декомпозиции размерности 2D и 3D указана в таблице 3. Для сравнения эффективности двух последних методов требуются дополнительные критерии.

Более того, практика показывает, что при шкалировании задачи с увеличением количества параллельных процессов в методе SDD, эффективность распараллеливания (отношение времени расчета на последовательной системе ко времени расчета на параллельной системе) деградирует даже при неизменном отношении $L^{\mathrm{BC}}/L^{\mathrm{in}}$ и объеме пересылок заведомо меньшем общей пропускной способности среды коммуникаций. Это является следствием фактического простоя части процессов в точках синхронизации с соседями (моменты синхронизации показаны на рис. 3 пунктиром и в методе SDD присутствуют на каждом временном шаге алгоритма) из-за случайного или регулярного отсутствия полной синхронности вычислений в различных процессах (неравномерность загрузки узлов, пиковые нагрузки в среде коммуникаций, ожидание прибытия граничных данных и т.п.). Оценить влияние этих явлений на эффективность распараллеливания, пользуясь лишь указанным критерием доли обменов, невозможно.

В связи с этим в работе предлагается другой критерий эффективной декомпозиции графа зависимостей, связанный с запасом асинхронности алгоритма — доля операций алгоритма, которые могут выполниться до момента синхронизации процесса с другими (вариант — полной синхронизации всех процессов задачи). Для ЯПФ вся задача синхронизируется на каждом ярусе, следовательно, если в алгоритме N_T шагов по времени, то запас его асинхронности будет $O(1/N_T)$ (как для одного процесса, так и для всей задачи). В случае SDD каждый шаг по времени происходит синхронизация соседних подобластей, а полная синхронизация задачи произойдет за N_P, $\sqrt{N_P}$ и $\sqrt[3]{N_P}$ шагов для 1D, 2D и 3D декомпозиции соответственно. Для LSTD моменты синхронизации определяются лишь информационными связями между операциями, что дает для полной синхронизации задачи N_z, N_y и N_x шагов для 1D, 2D и 3D декомпозиции соответственно. Оценка запаса асинхронности для различных методов указана в таблице 3.

Введение указанной величины позволяет оценить и эффективности распараллеливания задачи при ее шкалировании. В частности, если запас асинхронности алгоритма больше возможной несинхронности вычислений в различных процессах, то шкалирование будет происходить вообще без потери эффективности (то есть с эффективностью 100%). Учитывая реальные данные из таблицы 3, ясно, что для алгоритма LSTD шкалирование без потери эффективности на ПВС с достаточно большим количеством узлов является вполне возможным делом (см. результаты моделирования в разделе 6).

5 Рекурсивное разбиение графа зависимостей

Как неоднократно было указано ранее, рассмотренные алгоритмы имеют избыточный уровень внутреннего (логического) параллелизма (даже после проведения процедуры редукции пространственной размерности сетки КлА) и, следовательно, при их согласовании с реальным параллелизмом процессов ПВС (физическим) возможны различные стратегии дальнейшего укрупнения его уровня (зернистости).

Разработанный автором статьи метод LSTD предлагает стратегию, которая обеспечивает:

- автоматический учет многоуровневой иерархии подсистемы памяти современных компьютеров, которая включает регистровую память процессора, его кэш нескольких уровней, оперативную память, локальную дисковую память вычислительного узла и внешние сетевые запоминающие устройства;

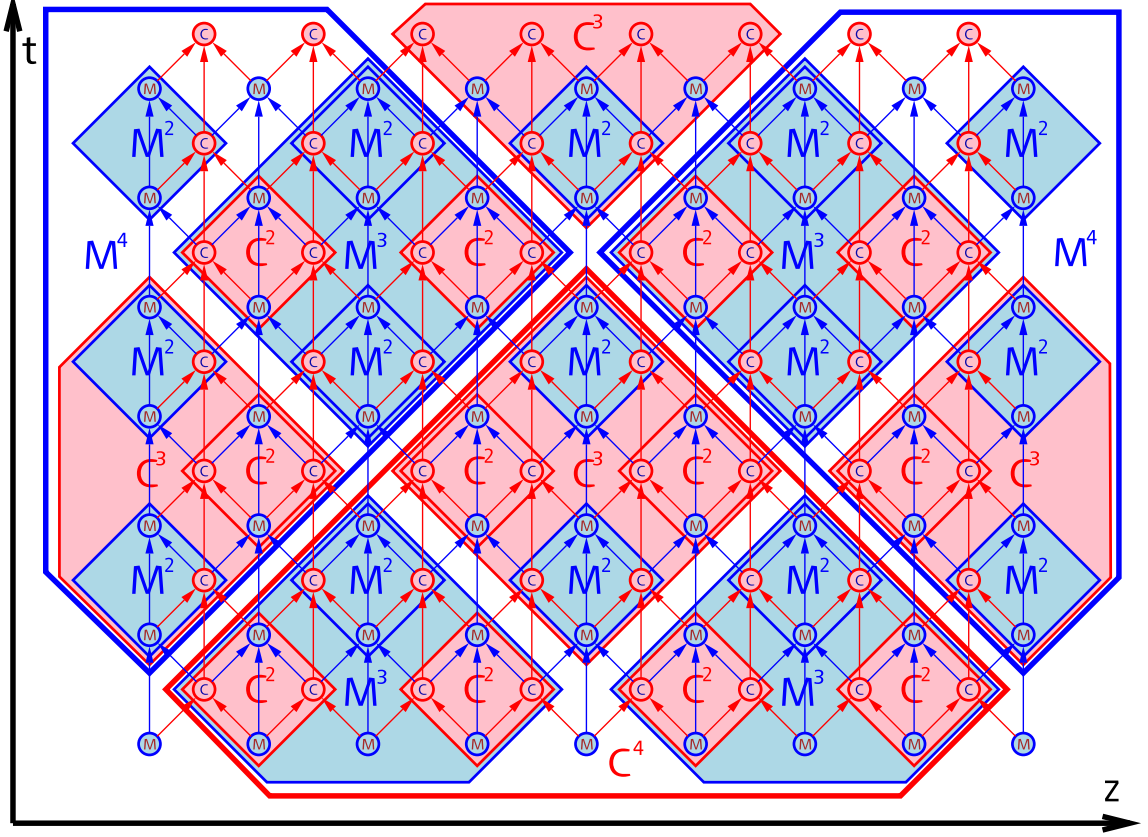

Рис. 4: Рекурсивная декомпозиция графа зависимостей, показанного на рис. 2 для 1D КлА. Показано 4 итерации рекурсии (уровня) — $(\mathtt{M},\mathtt{C}) \equiv (\mathtt{M}^1,\mathtt{C}^1), (\mathtt{M}^2,\mathtt{C}^2), (\mathtt{M}^3,\mathtt{C}^3), (\mathtt{M}^4,\mathtt{C}^4)$.

- средства и алгоритмы асинхронного управления вычислениями для обеспечения высокого запаса асинхронности и организации балансировки загрузки вычислительных узлов (статической, квазидинамической и динамической).

Выполнение первого пункта связано с локализацией данных алгоритма в верхних (самых быстрых) уровнях иерархии памяти. Иными словами, вычисления как можно дольше должны проводиться над тем объемом данных, который помещается в быструю память. В терминах информационного графа зависимостей, это требование соответствует укрупнению (композиции) тесно связанных узлов (операций) в гиперузлы, укрупнению граничных для гиперузлов ребер (зависимостей) в гиперребра. При этом результирующий (состоящий уже из гиперузлов) граф должен по-прежнему быть направленным ациклическим графом. В данной работе, дополнительно к этому, результирующий граф (гиперграф) рекурсивно подобен (сохраняет ту же структуру связей) исходному при бинарном укрупнении сетки КлА. Иными словами (случай 3D, другие размерности аналогично), для каждого из чисел $s = 1, \ldots n$ укрупнением исходной сетки КлА $N_x \times N_y \times N_z$ (приведена в разделе 3) в 2^s раз, построена сетка КлА: $N_x/2^s \times N_s/2^s \times N_s/2^s$ той же структуры, что исходная. Соответствующий граф зависимостей Γ_s имеет ту же структуру, что и графы зависимостей $\Gamma_{s-1}, \Gamma_{s-2}, \ldots \Gamma_0$, и состоит из узлов, являющихся композицией узлов Γ_{s-1}. Иллюстрация композиции в случае 1D приведена на рис. 4, откуда видно, что каждый из гиперузлов укрупненного графа зависимостей состоит из четырех соседних узла исходного графа (два `Cell` одного яруса и по одному `Mesh` с соседних ярусов). Аналогичным образом, ребра графа, соединяющие узлы, объединяются в гиперсвязи.

	i	$2k-1$	$2k$	$2k+1$	$2k+2$
$\mathtt{C}^{k,s+1}_{i_y,i_z}$	$(2i_y-\frac{1}{2},2i_z-\frac{1}{2})$	$\mathtt{M}^s_i \quad \mathtt{U}^{4\mathtt{S}^2}{}_i$	$\mathtt{U}^{4\mathtt{C}}{}_i \quad \mathtt{U}^{4\mathtt{S}^1}{}_i \quad \mathtt{M}^s_i$		
$\mathtt{S}^{1}{}^{k,s+1}_{\mathtt{z}i_y,i_z+\frac{1}{2}}$	$(2i_y+\frac{1}{2},2i_z-\frac{1}{2})$		$\mathtt{U}^{2\mathtt{S}^1_\mathtt{z}}{}_i \quad \mathtt{M}^s_i \quad \mathtt{U}^{2\mathtt{S}^2_\mathtt{y}}{}_i$		
$\mathtt{S}^{1}{}^{k,s+1}_{\mathtt{y}i_y+\frac{1}{2},i_z}$	$(2i_y-\frac{1}{2},2i_z+\frac{1}{2})$		$\mathtt{U}^{2\mathtt{S}^1_\mathtt{y}}{}_i \quad \mathtt{M}^s_i \quad \mathtt{U}^{2\mathtt{S}^2_\mathtt{z}}{}_i$		
$\mathtt{M}^{k,s+1}_{i_y+\frac{1}{2},i_z+\frac{1}{2}}$	$(2i_y+\frac{1}{2},2i_z+\frac{1}{2})$		$\mathtt{M}^s_i \quad \mathtt{U}^{4\mathtt{S}^2}{}_i$	$\mathtt{U}^{4\mathtt{C}}{}_i \quad \mathtt{U}^{4\mathtt{S}^1}{}_i \quad \mathtt{M}^s_i$	
$\mathtt{S}^{2}{}^{k,s+1}_{\mathtt{z}i_y,i_z+\frac{1}{2}}$	$(2i_y-\frac{1}{2},2i_z+\frac{1}{2})$			$\mathtt{U}^{2\mathtt{S}^1_\mathtt{z}}{}_i \quad \mathtt{M}^s_i \quad \mathtt{U}^{2\mathtt{S}^2_\mathtt{y}}{}_i$	
$\mathtt{S}^{2}{}^{k,s+1}_{\mathtt{y}i_y+\frac{1}{2},i_z}$	$(2i_y+\frac{1}{2},2i_z-\frac{1}{2})$			$\mathtt{U}^{2\mathtt{S}^1_\mathtt{y}}{}_i \quad \mathtt{M}^s_i \quad \mathtt{U}^{2\mathtt{S}^2_\mathtt{z}}{}_i$	
$\mathtt{C}^{k+1,s+1}_{i_y,i_z}$	$(2i_y-\frac{1}{2},2i_z-\frac{1}{2})$			$\mathtt{M}^s_i \quad \mathtt{U}^{4\mathtt{S}^2}{}_i$	$\mathtt{U}^{4\mathtt{C}}{}_i \quad \mathtt{U}^{4\mathtt{S}^1}{}_i \quad \mathtt{M}^s_i$

обозначения:
$$\mathtt{U}^{4\mathtt{C}}{}_{2i_y-\frac{1}{2},2i_z-\frac{1}{2}} = \mathtt{C}^s_{2i_y-1,2i_z-1} \cup \mathtt{C}^s_{2i_y,2i_z-1} \cup \mathtt{C}^s_{2i_y-1,2i_z} \cup \mathtt{C}^s_{2i_y,2i_z}$$
$$\mathtt{U}^{4\mathtt{S}^{1,2}}{}_{2i_y-\frac{1}{2},2i_z-\frac{1}{2}} = \mathtt{S}^{1,2}_\mathtt{y}{}^s_{2i_y-\frac{1}{2},2i_z-1} \cup \mathtt{S}^{1,2}_\mathtt{y}{}^s_{2i_y-\frac{1}{2},2i_z} \cup \mathtt{S}^{1,2}_\mathtt{z}{}^s_{2i_y-1,2i_z-\frac{1}{2}} \cup \mathtt{S}^{1,2}_\mathtt{z}{}^s_{2i_y,2i_z-\frac{1}{2}}$$
$$\mathtt{U}^{2\mathtt{S}^{1,2}_\mathtt{y}}{}_{2i_y-\frac{1}{2},2i_z-\frac{1}{2}} = \mathtt{S}^{1,2}_\mathtt{y}{}^s_{2i_y-\frac{1}{2},2i_z-1} \cup \mathtt{S}^{1,2}_\mathtt{y}{}^s_{2i_y-\frac{1}{2},2i_z}$$
$$\mathtt{U}^{2\mathtt{S}^{1,2}_\mathtt{z}}{}_{2i_y-\frac{1}{2},2i_z-\frac{1}{2}} = \mathtt{S}^{1,2}_\mathtt{z}{}^s_{2i_y-1,2i_z-\frac{1}{2}} \cup \mathtt{S}^{1,2}_\mathtt{z}{}^s_{2i_y,2i_z-\frac{1}{2}}$$

Таблица 4: 2D: Формулы для композиции узлов графа зависимостей уровня s (колонки 3-6) в гиперузлы графа зависимостей уровня $s+1$ (первая колонка), над колонками 3-6 подписаны номера кадров (ярусов параллельной формы графа)

Соответствующие формулы композиции: $s = 1, \dots n$; $\forall s$: $i_z = 1, \dots N_z/2^s$.

$$\mathtt{M}^{k,s+1}_{i_z+\frac{1}{2}} : \mathtt{M}^{2k-1,s}_{2i_z+\frac{1}{2}} \cup \left(\mathtt{C}^{2k-1,s}_{2i_z}\mathtt{C}^{2k-1,s}_{2i_z+1}\right) \cup \mathtt{M}^{2k,s}_{2i_z+\frac{1}{2}}$$
$$\mathtt{C}^{k,s+1}_{i_z} : \mathtt{M}^{2k,s}_{2i_z-\frac{1}{2}} \cup \left(\mathtt{C}^{2k,s}_{2i_z-1}\mathtt{C}^{2k,s}_{2i_z}\right) \cup \mathtt{M}^{2k+1,s}_{2i_z-\frac{1}{2}}$$

задают рекурсивное укрупнение графа при сохранении его исходной структуры.

Из данной стратегии укрупнения следует также дополнительное условие на размеры сетки КлА: они должны быть кратны 2^n. Число n выбирается таким образом, чтобы суммарный размер данных, соответствующих гиперузлу графа Γ_n превышал размер быстрой памяти вычислительного узла (обычно — размер кэша нижнего уровня, на данный момент порядка мегабайт), но не превышал размер медленной памяти (обычно — размер RAM, гигабайты). В этом случае локализация данных во всех уровнях иерархии быстрой памяти будет производиться автоматически. Ограничение на n сверху связано с требованиями, предъявляемыми алгоритмами балансировки загрузки.

Формулы для 2D и 3D композиции приведены в таблицах 4 и 5.

	$2k-1$	$2k$	$2k+1$	$2k+2$
$\mathtt{C}^{k,s+1}_{i_x,i_y,i_z}$	$\mathtt{M}^s_i \;\; \mathtt{U}^{6\mathtt{E}^2}{}_i \;\; \mathtt{U}^{12\mathtt{S}^2}{}_i$	$\mathtt{U}^{8\mathtt{C}}{}_i \;\; \mathtt{U}^{12\mathtt{S}^1}{}_i \;\; \mathtt{U}^{6\mathtt{E}^1}{}_i \;\; \mathtt{M}^s_i$		
$\mathtt{S}^{1}{}^{k,s+1}_{\mathtt{x}i_x+\frac12,i_y,i_z}\Big\vert_{\text{cicle}}$		$\mathtt{U}^{4\mathtt{S}^1_{\mathtt{x}}}{}_i \;\; \mathtt{U}^{6\mathtt{E}^1}{}_i \;\; \mathtt{M}^s_i \;\; \mathtt{U}^{2\mathtt{E}^2_{\mathtt{x}}}{}_i$		
$\mathtt{E}^{1}{}^{k,s+1}_{\mathtt{x}i_x,i_y+\frac12,i_z+\frac12}\Big\vert_{\text{cicle}}$		$\mathtt{M}^s_i \;\; \mathtt{U}^{4\mathtt{E}^2_{\mathtt{x}}}{}_i \;\; \mathtt{U}^{4\mathtt{S}^2_{\mathtt{x}}}{}_i$		
$\mathtt{M}^{k,s+1}_{i_x+\frac12,i_y+\frac12,i_z+\frac12}$		$\mathtt{M}^s_i \;\; \mathtt{U}^{6\mathtt{E}^2}{}_i \;\; \mathtt{U}^{12\mathtt{S}^2}{}_i$	$\mathtt{U}^{8\mathtt{C}}{}_i \;\; \mathtt{U}^{12\mathtt{S}^1}{}_i \;\; \mathtt{M}^s_i$	
$\mathtt{E}^{2}{}^{k,s+1}_{\mathtt{x}i_x+\frac12,i_y,i_z}\Big\vert_{\text{cicle}}$			$\mathtt{U}^{4\mathtt{S}^1_{\mathtt{y}}}{}_i \;\; \mathtt{M}^s_i \;\; \mathtt{U}^{2\mathtt{E}^2_{\mathtt{z}}}{}_i$	
$\mathtt{S}^{2}{}^{k,s+1}_{\mathtt{x}i_x+\frac12,i_y,i_z}\Big\vert_{\text{cicle}}$			$\mathtt{M}^s_i \;\; \mathtt{U}^{4\mathtt{E}^2_{\mathtt{z}}}{}_i \;\; \mathtt{U}^{4\mathtt{S}^2_{\mathtt{y}}}{}_i$	

обозначения: $i = i_x - \frac12, i_y - \frac12, i_z - \frac12$

$$\mathtt{U}^{8\mathtt{C}}{}_i = \left\{ \mathtt{C}^s_{i_x-d_x,i_y-d_y,i_z-d_z}\big|_{d_x,d_y,d_z\in\{0,1\}} \right\} \qquad \mathtt{U}^{4\mathtt{E}^{1,2}_{\mathtt{x}}}{}_i = \mathtt{U}^{2\mathtt{E}^{1,2}_{\mathtt{y}}}{}_i \cup \mathtt{U}^{2\mathtt{E}^{1,2}_{\mathtt{z}}}{}_i\Big|_{\text{cicle}}$$

$$\mathtt{U}^{4\mathtt{S}^{1,2}_{\mathtt{x}}}{}_i = \left\{ \mathtt{S}^{1,2}{}^s_{\mathtt{x}\,i_x-\frac12,i_y-d_y,i_z-d_z}\big|_{d_y,d_z\in\{0,1\}} \right\}_{\text{cicle}} \qquad \mathtt{U}^{12\mathtt{S}^{1,2}}{}_i = \mathtt{U}^{4\mathtt{S}^{1,2}_{\mathtt{x}}}{}_i \cup \mathtt{U}^{4\mathtt{S}^{1,2}_{\mathtt{y}}}{}_i \cup \mathtt{U}^{4\mathtt{S}^{1,2}_{\mathtt{z}}}{}_i$$

$$\mathtt{U}^{2\mathtt{E}^{1,2}_{\mathtt{x}}}{}_i = \left\{ \mathtt{E}^{1,2}{}^s_{\mathtt{x}\,i_x-d_x,i_y-\frac12,i_z-\frac12}\big|_{d_x\in\{0,1\}} \right\}_{\text{cicle}} \qquad \mathtt{U}^{6\mathtt{E}^{1,2}}{}_i = \mathtt{U}^{2\mathtt{E}^{1,2}_{\mathtt{x}}}{}_i \cup \mathtt{U}^{2\mathtt{E}^{1,2}_{\mathtt{y}}}{}_i \cup \mathtt{U}^{2\mathtt{E}^{1,2}_{\mathtt{z}}}{}_i$$

Таблица 5: Схема 3D композиции. Для простоты указаны только x-компоненты, формулы для y и z можно получить циклической перестановкой координат (отметка $|_{\text{cicle}}$)

6 Конечно-разностный метод временной области (FDTD) для решения системы уравнений Максвелла. Пример

Основные уравнения В качестве модельной задачи, иллюстрирующей возможности предложенного метода, рассмотрим решение при $t > 0$ системы уравнений Максвелла, описывающей эволюцию электромагнитных полей:

$$\frac{1}{c}\frac{\partial \vec{B}}{\partial t} = -\nabla \times \vec{E}, \qquad \frac{1}{c}\frac{\partial \vec{E}}{\partial t} = \nabla \times \vec{B} - \frac{4\pi}{c}\vec{j}, \tag{4}$$

$$\nabla \cdot \vec{E} = 4\pi\rho, \qquad \nabla \cdot \vec{B} = 0 \tag{5}$$

(c — скорость света в вакууме) с заданными начальными условиями на поля $\vec{E}(\vec{r}, t \le 0), \vec{B}(\vec{r}, t \le 0)$ и заданными во все моменты времени значениями плотностей зарядов и токов $\rho(\vec{r}, t), \vec{j}(\vec{r}, t)$ с учетом выполнения уравнения непрерывности (закон сохранения заряда):

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \vec{j} = 0. \tag{6}$$

Численный метод При решении будем использовать конечно–разностный метод временной области (FDTD) [12]. Следует отметить, что аналогичная постановка весьма востребованна в реальных расчетах широкого круга электромагнитных систем (см., например, [13, 14]).

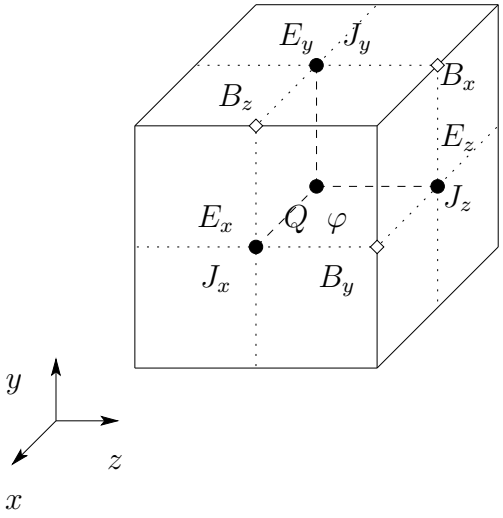

Рис. 5: Дискретные плотность заряда Q и тока $\vec{J} = (J_x, J_y, J_z)$, потенциал φ, электромагнитные поля $\vec{E} = (E_x, E_y, E_z)$, $\vec{B} = (B_x, B_y, B_z)$, определенные на сдвинутых сетках (ячейка Йи)

Запишем на однородной сетке G_3 центрированную по всем переменным конечно–разностную аппроксимацию дифференциальных уравнений (4)-(6):

$$\frac{1}{c}\frac{\vec{E}^k - \vec{E}^{k-1}}{\Delta^t} = \tilde{\nabla} \times \vec{B}^{k-\frac{1}{2}} - \frac{4\pi}{c}\vec{J}^{k-\frac{1}{2}}, \qquad \frac{1}{c}\frac{\vec{B}^{k+\frac{1}{2}} - \vec{B}^{k-\frac{1}{2}}}{\Delta^t} = -\tilde{\nabla} \times \vec{E}^k, \tag{7}$$

$$\tilde{\nabla} \cdot \vec{E}^k = 4\pi Q^k, \qquad \tilde{\nabla} \cdot \vec{B}^{k-\frac{1}{2}} = 0, \tag{8}$$

$$\frac{Q^k - Q^{k-1}}{\Delta^t} + \tilde{\nabla} \cdot \vec{J}^{k-\frac{1}{2}} = 0.$$

Здесь Δ^t — шаг по времени, дискретные значения электромагнитных полей $\vec{E}^k, \vec{B}^{k-\frac{1}{2}}$ и плотностей заряда Q^k и тока $\vec{J}^{k-\frac{1}{2}}$ определены на сдвинутых сетках в соответствии с рис. (5) в целые и полуцелые моменты времени соответственно, а $\tilde{\nabla}\times$ и $\tilde{\nabla}\cdot$ — центрированная конечно–разностная аппроксимация ротора и дивергенции соответственно, то есть ($\Delta^x, \Delta^y, \Delta^z$ — шаги сетки):

$$\left(\tilde{\nabla} \times \vec{F}\right)_{i_x,i_y,i_z} = \begin{pmatrix} \partial_y F_z|_{i_x,i_y,i_z} - \partial_z F_y|_{i_x,i_y,i_z} \\ \partial_z F_x|_{i_x,i_y,i_z} - \partial_x F_z|_{i_x,i_y,i_z} \\ \partial_y F_x|_{i_x,i_y,i_z} - \partial_x F_y|_{i_x,i_y,i_z} \end{pmatrix}$$

$$\left(\tilde{\nabla} \cdot \vec{F}\right)_{i_x,i_y,i_z} = \partial_x F_x|_{i_x,i_y,i_z} + \partial_y F_y|_{i_x,i_y,i_z} + \partial_z F_z|_{i_x,i_y,i_z}$$

$$\partial_x F|_{i_x,i_y,i_z} = \left(F_{i_x+\frac{1}{2},i_y,i_z} - F_{i_x-\frac{1}{2},i_y,i_z}\right)/\Delta^x,$$

$$\partial_y F|_{i_x,i_y,i_z} = \left(F_{i_x,i_y+\frac{1}{2},i_z} - F_{i_x,i_y-\frac{1}{2},i_z}\right)/\Delta^y,$$

$$\partial_z F|_{i_x,i_y,i_z} = \left(F_{i_x,i_y,i_z+\frac{1}{2}} - F_{i_x,i_y,i_z-\frac{1}{2}}\right)/\Delta^z$$

$$Q^k = \rho(\vec{r}_{i_x,i_y,i_z}, t_k)$$

$$\left\{J_x^{k-\frac{1}{2}}, J_y^{k-\frac{1}{2}}, J_z^{k-\frac{1}{2}}\right\} = \left\{j_x(\vec{r}_{i_x\pm\frac{1}{2},i_y,i_z}, t_{k-\frac{1}{2}}), j_y(\vec{r}_{i_x,i_y\pm\frac{1}{2},i_z}, t_{k-\frac{1}{2}}), j_z(\vec{r}_{i_x,i_y,i_z\pm\frac{1}{2}}, t_{k-\frac{1}{2}})\right\}$$

Подстановкой можно проверить, что уравнения (8) выполняются тождественно в любой момент времени, если они выполняются для начальных и граничных условий. Учитывая этот факт, а также то, что реальные вычисления проводятся с конечной точностью (связано с ошибкой округления), в дальнейшем достаточно рассматривать только решение эволюционных уравнений (7), точность при этом контролируется при помощи невязок:

$$E^k_{\text{err}} = \tilde{\nabla} \cdot \vec{E}^k - 4\pi Q^k, \qquad B^{k-\frac{1}{2}}_{\text{err}} = \tilde{\nabla} \cdot \vec{B}^{k-\frac{1}{2}}. \tag{9}$$

Данная схема имеет второй порядок точности по времени и всем координатам и устойчива при выполнении условия Куранта: $c\Delta^t \leq \min(\Delta^x, \Delta^y, \Delta^z)$.

Сведем в таблицу 6 формулы, связывающие решение дискретной задачи (7), (9) с введенными ранее КлА.

7 Результаты и обсуждение

Вычислительные ресурсы Вычислительные проблемы при решении данной системы уравнений связаны с тем, что требуется разрешить на сетке самый мелкий пространственный масштаб изменения электромагнитного поля. Так как поле имеет волновую природу, то пространственный шаг сетки обычно выбирают равным десятой доли самой маленькой длины электромагнитной волны из всех возможных для данной физической ситуации. Из-за этого в реальных задачах подробной сеткой приходится покрывать большие пространственные области. Это приводит к большому количеству данных, обрабатываемых на каждом шаге по времени. Если данные не помещаются в быстрой памяти, доступной одному процессору, то параллельные вычисления становятся необходимы. Кроме того, из-за условия Куранта, шаг по времени также достаточно мал, что приводит к большому числу требуемых при расчете временных шагов и, следовательно, больших вычислительных ресурсов компьютера.

На каждом шаге по времени алгоритм имеет линейную вычислительную сложность по данным. Алгоритм включает 66 элементарных операций (36 сложений и 30 умножений), не считая функций для вычисления плотностей зарядов и токов, кроме того, 36 доступов в память на чтение и 8 на запись в расчете на один шаг по времени на каждую ячейку сетки, а также постоянного хранения 8 чисел с плавающей точкой для каждой ячейки, не включая констант. Подобное соотношение числа операций к числу операндов при значительном их общем количестве, является сложным для современных компьютеров, так как последние не сбалансированы по соотношению производительность процессора/скорость обмена с памятью. Все это приводит к низкой эффективности вычислений по сравнению с теоретической производительностью процессора.

Тестовые системы Проведем тестирование алгоритма LSTD и его сравнение с обычными пошаговыми вычислениями на трех разных вычислительных системах. Целью такого выбора является демонстрация возможностей предложенного метода в преодолении трех основных ограничивающих факторов в задачах моделирования среды:

- размер оперативной памяти ограничивает размер задачи, а скорость доступа в нее — скорость расчета;

	Вектор состояния	Функция перехода
1D	$f^{C,k}: \begin{pmatrix} E_x^k, \\ E_y^k, \\ E_{\text{err}}^k \\ B_z^{k+\frac{1}{2}} \end{pmatrix}$	$\mathtt{C}: f^{C,k} = \begin{pmatrix} E_x^k + c\Delta^t(\partial_y B_z^{k-\frac{1}{2}} - \partial_z B_y^{k-\frac{1}{2}}) - 4\pi\Delta^t J_x^{k-\frac{1}{2}} \\ E_y^k + c\Delta^t(\partial_z B_x^{k-\frac{1}{2}} - \partial_x B_z^{k-\frac{1}{2}}) - 4\pi\Delta^t J_y^{k-\frac{1}{2}} \\ \partial_x E_x^k + \partial_y E_y^k + \partial_z E_z^k - 4\pi Q^k \\ B_z^{k-\frac{1}{2}} - c\Delta^t(\partial_y E_x^k - \partial_x E_y^k) \end{pmatrix}$
	$f^{M,k}: \begin{pmatrix} B_x^{k+\frac{1}{2}} \\ B_y^{k+\frac{1}{2}} \\ B_{\text{err}}^{k+\frac{1}{2}} \\ E_z^{k+1} \end{pmatrix}$	$\mathtt{M}: f^{M,k} = \begin{pmatrix} B_x^{k+\frac{1}{2}} - c\Delta^t(\partial_y E_z^k - \partial_z E_y^k) \\ B_y^{k+\frac{1}{2}} - c\Delta^t(\partial_z E_x^k - \partial_x E_z^k) \\ \partial_x B_x^{k+\frac{1}{2}} + \partial_y B_y^{k+\frac{1}{2}} + \partial_z B_z^{k+\frac{1}{2}} \\ E_z^k + c\Delta^t(\partial_y B_x^{k+\frac{1}{2}} - \partial_x B_y^{k+\frac{1}{2}}) - 4\pi\Delta^t J_z^{k+\frac{1}{2}} \end{pmatrix}$
2D	$f^{C,k}: \begin{pmatrix} E_x^k \\ E_{\text{err}}^k \end{pmatrix}$	$\mathtt{C}: f^{C,k} = \begin{pmatrix} E_x^{k-1} + c\Delta^t(\partial_y B_z^{k-\frac{1}{2}} - \partial_z B_y^{k-\frac{1}{2}}) - 4\pi\Delta^t J_x^{k-\frac{1}{2}} \\ \partial_x E_x^k + \partial_y E_y^k + \partial_z E_z^k - 4\pi Q^k \end{pmatrix}$
	$f_y^{S,k}: \begin{pmatrix} B_z^{k+\frac{1}{2}} \\ E_y^{k+1} \end{pmatrix}$	$\mathtt{S}_\mathtt{y}^\mathtt{1}: B_z^{k+\frac{1}{2}} = B_z^{k-\frac{1}{2}} - c\Delta^t(\partial_y E_x^k - \partial_x E_y^k)$
		$\mathtt{S}_\mathtt{y}^\mathtt{2}: E_y^{k+1} = E_y^k + c\Delta^t(\partial_z B_x^{k+\frac{1}{2}} - \partial_x B_z^{k+\frac{1}{2}}) - 4\pi\Delta^t J_y^{k-\frac{1}{2}}$
	$f_z^{S,k}: \begin{pmatrix} B_y^{k+\frac{1}{2}} \\ E_z^{k+1} \end{pmatrix}$	$\mathtt{S}_\mathtt{z}^\mathtt{1}: B_y^{k+\frac{1}{2}} = B_y^{k-\frac{1}{2}} - c\Delta^t(\partial_z E_x^k - \partial_x E_z^k)$
		$\mathtt{S}_\mathtt{z}^\mathtt{2}: E_z^{k+1} = E_z^k + c\Delta^t(\partial_y B_x^{k+\frac{1}{2}} - \partial_x B_y^{k+\frac{1}{2}}) - 4\pi\Delta^t J_z^{k-\frac{1}{2}}$
	$f^{M,k}: \begin{pmatrix} B_x^{k+\frac{1}{2}} \\ B_{\text{err}}^{k+\frac{1}{2}} \end{pmatrix}$	$\mathtt{M}: f^{M,k} = \begin{pmatrix} B_x^{k-\frac{1}{2}} - c\Delta^t(\partial_y E_z^k - \partial_z E_y^k) \\ \partial_x B_x^{k+\frac{1}{2}} + \partial_y B_y^{k+\frac{1}{2}} + \partial_z B_z^{k+\frac{1}{2}} \end{pmatrix}$
3D	$f^{C,k}: E_{\text{err}}^k$	$\mathtt{C}: f^{C,k} = \partial_x E_x^k + \partial_y E_y^k + \partial_z E_z^k - 4\pi Q^k$
	$f_x^{E,k}: B_x^{k+\frac{1}{2}}$	$\mathtt{E}_\mathtt{x}^\mathtt{1}: f_x^{E,k} = B_x^{k-\frac{1}{2}} - c\Delta^t(\partial_y E_z^k - \partial_z E_y^k)$
	$f_y^{E,k}: B_y^{k+\frac{1}{2}}$	$\mathtt{E}_\mathtt{y}^\mathtt{1}: f_y^{E,k} = B_y^{k-\frac{1}{2}} - c\Delta^t(\partial_z E_x^k - \partial_x E_z^k)$
	$f_z^{E,k}: B_z^{k+\frac{1}{2}}$	$\mathtt{E}_\mathtt{z}^\mathtt{1}: f_z^{E,k} = B_z^{k-\frac{1}{2}} - c\Delta^t(\partial_y E_x^k - \partial_x E_y^k)$
	$f^{M,k}: B_{\text{err}}^{k+\frac{1}{2}}$	$\mathtt{M}: f^{M,k} = \partial_x B_x^{k+\frac{1}{2}} + \partial_y B_y^{k+\frac{1}{2}} + \partial_z B_z^{k+\frac{1}{2}}$
	$f_x^{S,k}: E_x^k$	$\mathtt{S}_\mathtt{x}^\mathtt{2}: f_x^{S,k} = E_x^k + c\Delta^t(\partial_y B_z^{k-\frac{1}{2}} - \partial_z B_y^{k-\frac{1}{2}}) - 4\pi\Delta^t J_x^{k-\frac{1}{2}}$
	$f_y^{S,k}: E_y^k$	$\mathtt{S}_\mathtt{y}^\mathtt{2}: f_y^{S,k} = E_y^k + c\Delta^t(\partial_z B_x^{k-\frac{1}{2}} - \partial_x B_z^{k-\frac{1}{2}}) - 4\pi\Delta^t J_y^{k-\frac{1}{2}}$
	$f_z^{S,k}: E_z^k$	$\mathtt{S}_\mathtt{z}^\mathtt{2}: f_z^{S,k} = E_z^k + c\Delta^t(\partial_y B_x^{k-\frac{1}{2}} - \partial_x B_y^{k-\frac{1}{2}}) - 4\pi\Delta^t J_z^{k-\frac{1}{2}}$
		$\mathtt{S}_\mathtt{x}^\mathtt{1}$, $\mathtt{S}_\mathtt{y}^\mathtt{1}$, $\mathtt{S}_\mathtt{z}^\mathtt{1}$, $\mathtt{E}_\mathtt{x}^\mathtt{2}$, $\mathtt{E}_\mathtt{y}^\mathtt{2}$, $\mathtt{E}_\mathtt{z}^\mathtt{2}$ — "пустые" преобразования

Таблица 6: КлА для алгоритма FDTD

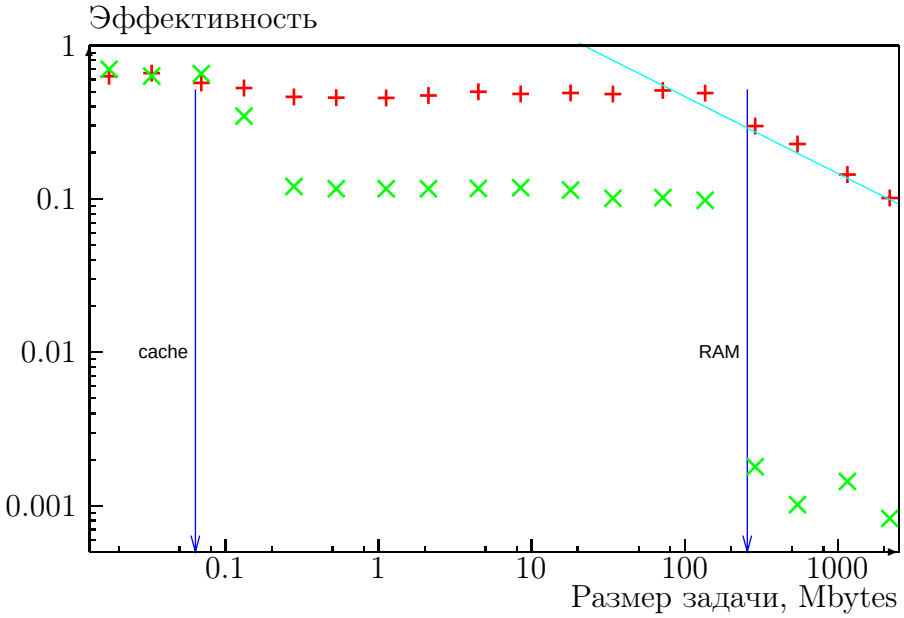

Рис. 6: Серия расчетов S1. Эффективность алгоритма LSTD (+) при изменении размера данных задачи в сравнении с обычными пошаговыми вычислениями (×). Значение эффективности 1 соответствует теоретической производительности процессора (может быть выполнено не более 1 сложения и 1 умножения за такт). Вертикальными линиями отмечены размеры данных, соответствующие размерам кэша и оперативной памяти. Интерполяционная линия соответствует $\sqrt{N_{\text{run}}}$

- самый медленный узел и латентность коммуникаций ограничивают эффективность распараллеливания;
- от вычислительной системы требуется высокая однородность.

Было проведено три серии расчетов, по одной на каждой из следующих вычислительных систем:

S1 "Обычный" малопроизводительный персональный компьютер со скромной оперативной памятью и дисковой подсистемой (Duron@600MHz с памятью SDR 256MB и ATA жестким диском).

S2 Наиболее мощный в России суперкомпьютер кластерной архитектуры (МВС 1000М — 384 узла, коммуникационная среда Myrinet, в каждом: 2 процессора Alpha21264@667MHz и 2GB оперативной памяти)

S3 Локальная вычислительная сеть лаборатории, 5 персональных компьютеров с вариацией частоты в 3 раза и размера памяти в 4 раза, среда коммуникаций Ethernet 100Mbit.

В обсуждаемых ниже расчетах был использован 2D вариант алгоритма LSTD, с двумерной декомпозицией трехмерной пространственной области расчета. В каждой из серий при одинаковых начальных и граничных условиях варьировались размеры сетки.

В серии S1 исследовались возможности алгоритма по автоматической локализации данных в верхнем уровне подсистемы памяти (в данном случае, внутри кэша L1 процессора размером 64KB) при изменении размера данных задачи N_{run} от 17KB (задача заведомо помещается в кэш целиком) до 2GB (размер данных задачи на порядок превосходит размер оперативной памяти компьютера).

Как видно из рисунка 6, обычный пошаговый алгоритм расчета ($\times$) ведет себя следующим образом: если данные помещаются в кэш целиком ($N_{\text{run}} \leq 64\text{KB}$, что с практической точки зрения не интересно), производительность близка к теоретической (превышает 50%). При увеличении размера задачи сверх удвоенного размера кэша производительность скачком падает в 5 раз (до уровня 10% от теоретического) и остается примерно на этом уровне пока размер задачи меньше размера доступной оперативной памяти $N_{\text{run}} \leq 240\text{MB}$ (с небольшой тенденцией к снижению из-за переполнения буферов дескрипторов страниц при $N_{\text{run}} \geq 64\text{MB}$). Этот уровень эффективности и считается обычным ("штатным") для указанного круга задач. Как только размер задачи превосходит размер доступной оперативной памяти, включается механизм подкачки страниц виртуальной памяти с жесткого диска, и производительность падает еще в 100 раз, что делает сам расчет данных такого размера неактуальным для данной вычислительной системы.

В то же время эффективность алгоритма LSTD ($+$) остается на высоком уровне (50-70%%) не только при помещающейся в кэш задаче, но и вплоть до размеров доступной оперативной памяти. И только при размерах задачи, существенно превышающих RAM, эффективность начинает снижаться (как корень из размера задачи). При этом вплоть до размеров, превышающих размер RAM на порядок, эффективность остается выше эффективности обычного пошагового алгоритма, используемого в "штатном" для него режиме.

Следует также отметить, что в данном случае не проводилось никаких дополнительных попыток оптимизации ни самой дисковой подсистемы, ни доступа к ней. Легко предположить, что использование более быстрого диска, RAID массива или специфически оптимизированного для алгоритма LSTD механизма виртуальной памяти, приведет к существенному уменьшению скорости потери эффективности вплоть до пренебрежимо малого значения.

Таким образом, использование алгоритма LSTD дает возможность проведения расчетов задач с размером данных, существенно превышающих размер оперативной памяти и с эффективностью, определяемой доступом в кэш–память процессора одновременно.

В серии S2 (рис. 7) исследовалась эффективности шкалирования при распараллеливании в однородных сетях (суперкомпьютер кластерного типа) на однородных задачах, не требующих динамической балансировки нагрузки. Под однородностью задачи здесь понимается одинаковое количество элементарных операций алгоритма, выполняющихся в каждом из MPI процессов. Подобная однородность характерна для конечно–разностных методов на регулярной постоянной сетке. Возможная несинхронность расчета на разных процессорах однородной сети (кластера) при этом связана обычно со случайными или регулярными факторами загрузки среды коммуникаций, ее топологией, индивидуальными настройками узлов кластера (вплоть до вариаций частоты тактовых генераторов). В предварительной серии расчетов было выяснено, что в среде тестируемой ПВС (МВС 1000М) указанная несинхронность носила в основном случайный характер и имела близкое к нормальному распределение с полушириной около 3%.

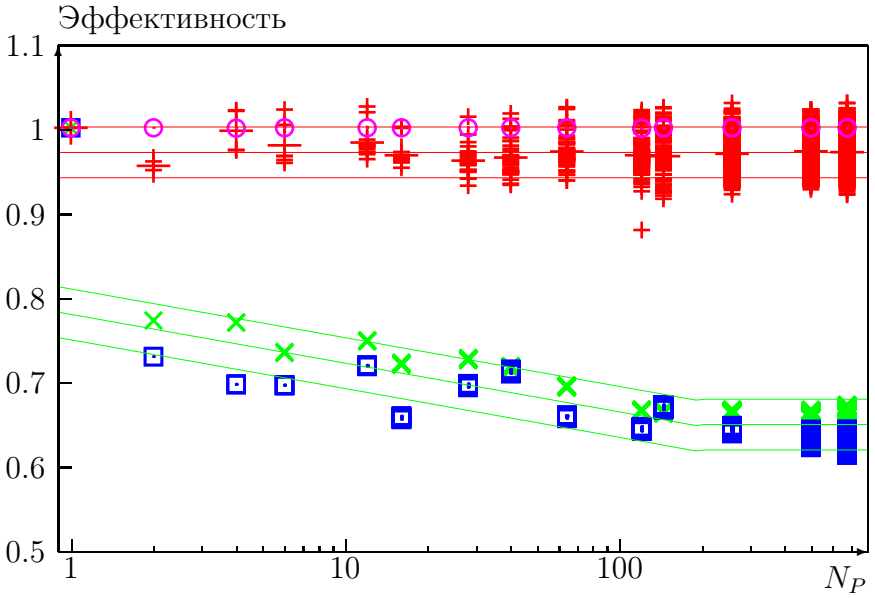

Рис. 7: Серия S2. Эффективность шкалирования при распараллеливании. Каждая точка на графике представляет момент окончания расчета (192 шага по времени) для одной подобласти (одного процесса). Методы: LSTD (+, межпроцессорный обмен 580 КБ/сек), и SDD (межпроцессорный обмен 22.8 КБ/сек × и 88 КБ/сек ▫). Параллельными линиями отмечены средние значения с учетом доверительного интервала

Эффективность шкалирования при распараллеливании рассчитывается как отношение времени параллельного расчета (сумма времен выполнения процессов деленное на их число) к времени последовательного расчета (расчета на однопроцессорной системе) тем же методом в зависимости от числа процессоров. Количество обрабатываемых данных и количество пересылаемых данных, приходящихся на один процессор при этом остается одинаковым. Это сделано для того, чтобы исследовать эффект изменения эффективности при распараллеливании “в чистом виде”, исключив влияние конкурирующих эффектов (изменение доли обменов и кэширования).

Размеры пространственной области расчета были выбраны так, чтобы доля обменов для обычного метода декомпозиции пространственной области (SDD) была мала, $\sim 10^{-4}$, или, в единицах пропускной способности каналов коммуникаций, $\simeq 23$ КБ/сек в расчете на каждый процессор для полученной скорости расчета (×). Это ничтожно мало по сравнению с заявленной ПС среды коммуникаций кластера [15] (Myrinet 2000, 250 МБ/сек). Более того, при тестировании методом LSTD (+), который за счет автоматической локализации вычислений в кэш–памяти процессора (смотри серию S1) имеет на порядок более высокую производительность, и, следовательно, на тот же порядок более высокие требования к ПС среды коммуникаций, оказалось, что реально, без потери эффективности, по сети можно передавать в 25 раз больше данных, чем требуется в методе SDD.

Для исследования влияния латентности коммуникаций, которая также может вносить свой вклад в снижение эффективности распараллеливания в методе SDD, была проведена контрольная серия расчетов тем же методом (▫) с повышенным в 4 раза количеством и объемом обменов. Из сравнения графиков видно, что это влияние для данных параметров несущественно.

Столь малая доля обменов определяет практически идеальные условия для метода SDD, который должен в таких условиях обеспечивать теоретическую 100% эффективность распараллеливания при шкалировании. Однако на соответствующих графиках наблюдается снижение эффективности на $> 10\%$ при увеличении числа двухпроцессорных узлов кластера от 1 до ста. Как указывалось ранее, это связано с тем, что на каждом временном шаге соседние подобласти должны синхронизироваться между собой, иными словами, запас асинхронности задачи $O(\sqrt{N_P}/N_T)$ в данном случае $\ll 1$. Из той же оценки следует, что при очень большом числе узлов (> 100), когда запас асинхронности задачи достигнет нескольких процентов и превзойдет уровень несинхронности расчетов на отдельных узлах, ситуация может несколько стабилизироваться, что также заметно на графиках. К сожалению для метода SDD, даже такой умеренный уровень запаса асинхронности является недостижимым в задачах реальной сложности при большом числе шагов по времени и/или при решении вычислительно неоднородных задач, что и предопределяет наблюдаемое в реальных расчетах снижение эффективности шкалирования при распараллеливании.

В случае же метода LSTD при тех же численных параметрах, что и в первой серии расчетов методом SDD, запас асинхронности задачи $O(N_y/N_T) \sim 500\sqrt{N_P}/N_T$ уже $\gg 1$ при любом N_P. Соответственно, причин снижения эффективности из-за требований синхронизации нет, что и демонстрируют результаты тестирования алгоритма LSTD, показывающие неизменную эффективность шкалирования (уровень 0.971 на графике) на любом количестве процессоров из имеющихся (до 700). Наблюдаемая же при малом числе процессоров (включая и однопроцессорный случай) вариация эффективности объясняется статистической погрешностью.

Для выяснения точного значения эффективности можно обратиться к выдаваемой при расчете статистике простоя процесса из-за ожидания данных. Оказалось, что из всей серии расчетов (в сумме 1861 процессов), простой наблюдался лишь 3 раза (на 0.07 сек, 0.13 сек и 4.5 сек, последнее значение и дало отдельный выброс на графике при $N_p = 120$). Даже с учетом указанных фактов посчитанная по статистике простоя процессоров эффективность (фиолетовые кружки) отличается от 100% только в 4 знаке.

Следует также заметить, что в данном случае та же 100% эффективность должна сохраняться и при дальнейшем увеличении количества процессоров, если только будет выполнены два относительно простых для современных ПВС условия:

- Сохранится баланс уровня несинхронности расчетов и уровня запаса асинхронности — и ПВС, и сама задача должны иметь уровень неоднородности меньший запаса асинхронности.

- Пропускная способность межпроцессорных коммуникаций для локальных связей в расчете на процессор не уменьшается с ростом числа процессоров.

Экстраполируя результаты данного тестирования, необходимо отметить, что они могут быть особенно интересны для ПВС следующего поколения, которые должны поддерживать несколько десятков–сотен тысяч параллельных процессов [16]. Для таких ПВС указанное принципиальное падение эффективности синхронных методов не позволит проводить расчеты эффективно, в то время как метод LSTD сохранит высокую эффективность.

В серии S3 были исследованы методы эффективного проведения расчетов в неоднородных сетях. При этом использовались различные алгоритмы статической, квазидинамической

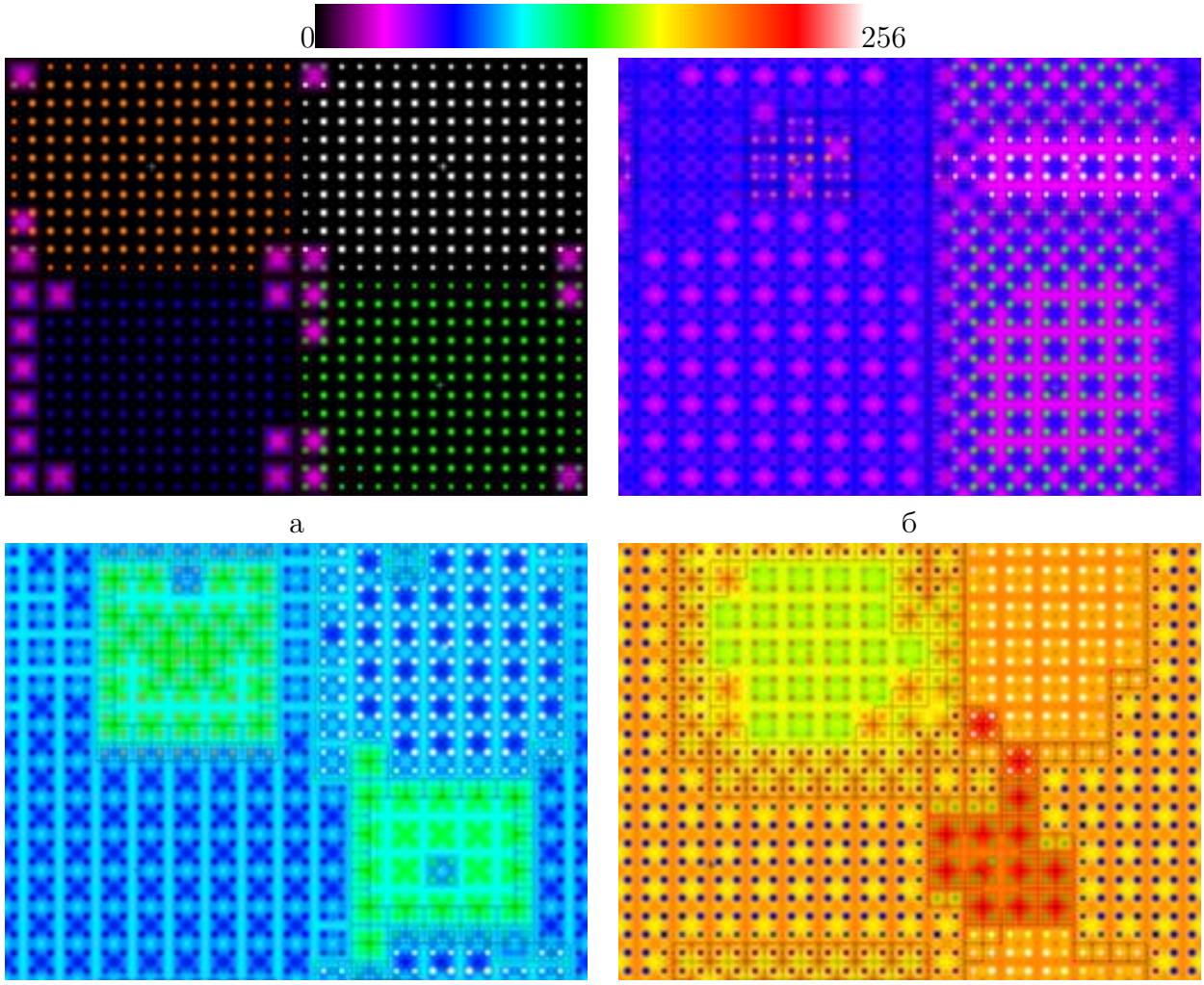

Рис. 8: Иллюстрация работы алгоритма динамической балансировки загрузки в LSTD. Сетка 1024×768, размер “кирпича” 32×32, всего “кирпичей” 32×24. Мгновенные снимки состояния ячеек сетки (локального шага по времени в соответствии со шкалой) при расчете в неоднородной ПВС для последовательных моментов времени (а—г). Принадлежность “кирпича” к процессу (всего 4 процессора, отличающихся по производительности в 4 раза) отмечено цветом квадратика в центре этого “кирпича”.

и динамической балансировки загрузки, базирующиеся на методе LSTD.

Перед началом расчета исходная сетка 2D КлА разбивается на непересекающиеся подобласти размером $2^n \times 2^n$ ячеек каждая (см. раздел 5), так называемые "кирпичи", которые в дальнейшем выступают единицей планирования в управлении данными. Каждый кирпич в каждый момент времени либо принадлежит какому-то конкретному процессу, либо пересылается от одного процесса к другому. Если четыре соседних (имеющих общий угол) кирпича принадлежат одному процессу, они образуют блок одного из четырех типов (в соответствии с описанными ранее типами КлА). Данный блок может порождать квант вычислений, являющийся пространственно–временной композицией элементарных операций в соответствии с описанным в разделе 5 алгоритмом. Условия порождения кванта вычислений — готовность зависимых данных.

Основной задачей алгоритмов управления является пересылка кирпичей между процессами таким образом, чтобы в любой момент времени в каждом из процессов было достаточно блоков, готовых породить квант вычислений. Иными словами, процессы должны быть постоянно заняты вычислениями.

В зависимости от степени неоднородности задачи можно использовать один из трех алгоритмов балансировки загрузки:

Статический: В соответствии с априорными данными о производительности процессоров исходная область разделяется на подобласти по числу процессоров, причем количество кирпичей в каждой из них пропорционально производительности соответствующего процессора. Граничные кирпичи при необходимости пересылаются между процессами, но вычисления проводятся всегда в том процессе, которому изначально принадлежал левый нижний кирпич блока.

Квазистатический: Первоначальное разбиение области как и в случае статической балансировки. Пересылаться также могут только граничные кирпичи, но направление пересылки и, соответственно, процесс, который и будет производить сами вычисления, выбирается, исходя из информации о текущем среднем локальном времени составляющих процесс кирпичей.

Динамический: Первоначальное разбиение области достаточно произвольно и не требует априорной информации, так как размеры подобластей динамически подстраиваются под доступные вычислительные ресурсы процессов. Геометрический центр каждой из подобластей и их среднее состояние отслеживаются динамически. Любые кирпичи могут пересылаться в соответствие с информацией о текущем среднем времени процессов, но приоритет в пересылке отдается кирпичам, максимально удаленным от центра. Необходимо отслеживать тупики и конфликты.

На рис.8 показана иллюстрация работы (четыре мгновенных снимка текущего состояния ячеек КлА) алгоритма динамической балансировки загрузки для ПВС, состоящей из четырех процессов, имеющих вариацию в производительности в 4 раза. Принадлежность кирпича к тому или иному процессу показана цветом квадратика в центре кирпича (если кирпич внутренний) или ромба (если он граничный или неопределенный). На рис.8(а) приведено состояние системы сразу после старта, при этом в начальный момент времени область была разделена поровну между четырьмя процессами. Как видно из рисунков (б) и (в), с течением времени

самый быстрый процесс (изначально занимавший левый нижний угол области) увеличивает размер обсчитываемой подобласти, в основном, за счет самого медленного процесса (изначально — левый верхний угол). Также видно, что ситуация меняется динамически в зависимости от текущей загрузки процессов, отдельные кирпичи эволюционируют асинхронно, однако вся система в среднем эволюционирует синхронно.

8 Заключение

На основе анализа тонкой информационной структуры алгоритмов, базирующихся на явных сеточных методах с локальным шаблоном, разработан метод локальной пространственно–временной декомпозиции. В методе используется рекурсивная композиция/декомпозиция параллельной формы графа зависимостей, укрупняющая избыточный мелкозернистый параллелизм в целях повышения локальности и создания запаса асинхронности вычислений на уровне среднезернистого параллелизма.

Для удобства описания информационных связей использована методология асинхронно управляемых клеточных автоматов. Разрабатываются алгоритмы децентрализованного асинхронного управления вычислениями, при этом используются те же структурные элементы и их связи, что и при декомпозиции графа зависимостей.

В качестве примера алгоритм был применен для конечно–разностного метода временной области, использованного при решении системы уравнений Максвелла. Результаты тестирования алгоритма при шкалировании задачи по размеру данных показали:

- за счет автоматической локализации данных на верхнем уровне иерархии подсистемы памяти компьютера, эффективность расчетов приближается к теоретическому пределу (составляет более 50%) для размера данных задачи, сравнимого с размером оперативной памяти;
- вплоть до размера задачи, на порядок превышающей размер оперативной памяти, эффективность алгоритма превышает эффективность обычных пошаговых алгоритмов, работающих “в штатном” для себя режиме;
- при шкалировании задачи достигается 100% эффективность распараллеливания при любом числе процессоров;
- построенные на базе алгоритма методы статической и динамической балансировки загрузки позволяют эффективно проводить расчеты в неоднородных параллельных вычислительных системах.

9 Благодарности

Автор выражает благодарность Левченко Татьяне Викторовне за полезные обсуждения и ценные замечания. Работа выполнена при частичной поддержке гранта РФФИ 02-01-01004 и программы ОМН РАН №3.

Список литературы

[1] *Якобовский М.В.* Распределенные вычисления. — Москва: Наука, 2003.

[2] *Воеводин В.В.* Математические модели и методы в параллельных процессах. — Москва: Наука, 1986.

[3] *Kuzmin D.A., Kazakov F.A., Legalov A.I.* Description of parallel-functional programming language // Advances in Modeling & Analysis. AMSE Press. — 1995. — Т. 28, № 3. — С. 1–17. см. также http://www.softcraft.ru/parallel/fpp/fppcontent.shtml.

[4] *Wolfram S.* Cellular automata as models for complexity // Nature. — 1984. — Т. 311. — С. 419.

[5] *Achasova S., Bandman O., Markova V., Piskunov S.* Parallel Substitution Algorithm: Theory and Application. — Singapore et al.: World Scientific, 1994.

[6] *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. — СПб.: БХВ-Петербург, 2004.

[7] *Самарский А.А., Попов Ю.П.* Разностные методы решения задач газовой динамики. — Москва: Наука, 1980.

[8] *Hendrickson Bruce, Kolda Tamara G.* Graph partitioning models for parallel computing // Parallel Computing. — 2002. — Т. 26, № 12.

[9] *Gropp William D., Keyes David E.* Domain decomposition on parallel computers // Impact Comput. Sci. Eng.. — 1989. — Т. 1. — С. 421–439.

[10] *Levchenko V.D.* // Proceedings of 16th Int. Conf. on the Numerical Simulation of Plasmas (ICNSP-98). — Santa-Barbara, California, USA. February 10-12, 1998. — С. 147–150.

[11] *Ёлкина Н.В., Левченко В.Д.* // Труды Всероссийской конференции "Высокопроизводительные вычисления и их приложения". — Черноголовка. — 2000. : Издательство московского университета. — С. 278–281.

[12] *Yee K.S.* Numerical solution of initial boundary value problems including maxwell's equations in isotropic media // IEEE Trans. Antennas & Propagation. — 1966. — Т. 14. — С. 302–307.

[13] *Tokushima M., Yamada H.* Light propagation in a photonic-crystal-slab line-defect wave guide // IEEE JQE. — 2002. — Т. 38, № 7. — С. 753–759.

[14] *Oguz U., Gurel L.* Frequency responses of ground-penetrating radars operating over highly lossy grounds // IEEE Tran. On Geoscience and Remote Sensing. — 2002. — Т. 40, № 6. — С. 1385–1394.

[15] http://www.jscc.ru/scomputer.html.

[16] U.s. department of energy's office of science unveils 20-year vision for the future of basic research. — February 12, 2004.