УДК 004.032.24+004.451.23+004.451.34

РАСПАРАЛЛЕЛИВАНИЕ И ОПТИМИЗАЦИЯ ПОСТРОЕНИЯ БЛОЧНЫХ РАСЧЕТНЫХ СЕТОК В ПРЕПРОЦЕССОРЕ ПАКЕТА ПРОГРАММ "ЛОГОС"

В. В. Лазарев
(ФГУП "РФЯЦ-ВНИИЭФ", г. Саров Нижегородской области)

Рассматриваются способы, оптимизирующие построение блочных расчетных сеток по подготовленной декомпозиции геометрической модели на блоки. Описаны структуры данных, представляющие декомпозицию и блочные сетки. На основе этих представлений описаны распараллеленные алгоритмы построения блочной сетки. Приведены графики зависимостей эффективности распараллеливания и коэффициента ускорения от количества потоков. Разработан также алгоритм, позволяющий перестраивать сетку по измененной декомпозиции только на измененных блоках. Для сеток, не помещающихся в оперативную память, предложены пофрагментные построение и запись в отдельные файлы. Построена сетка, содержащая 1 млрд ячеек. Алгоритмы реализованы и внедрены в препроцессор ЛОГОС, начиная с версии 5.1.

Ключевые слова: распараллеливание, OpenMP, блочная расчетная сетка, декомпозиция геометрии на блоки, препроцессор пакета программ ЛОГОС.

Введение

Построение блочных сеток [1] в препроцессоре пакета программ ЛОГОС [2] основывается на последовательном формировании связанных блоков вокруг геометрической модели. Назовем этот процесс декомпозицией, которая подробно описана в [3]. На основе сформированных блоков строится расчетная сетка. Декомпозиция геометрической модели в препроцессоре выполняется пользователем вручную в интерактивном режиме. Чем сложнее исходная геометрическая модель, тем дольше выполняется декомпозиция. Сложность также связана с ограничением на форму получаемых блоков: они должны быть шестигранными или четырехугольными для возможности построения на них структурированных сеток. Для оценки промежуточных результатов пользователь запускает процедуру построения сетки на сформированных блоках. Частые итерации задают ограничение на время перестроения блочной сетки: чем оно меньше, тем лучше. Способам решения этой задачи посвящена статья.

Известны аналогичные работы, выполненные в других системах построения расчетных сеток. Распараллеливание построения сеток по независимым фрагментам реализовано в программном продукте Ansys Mechanical, начиная с версии 15.0 [4]. В библиотеке Pamgen из проекта Trilinos [5] реализовано построение расчетных сеток на простых геометриях посредством их декомпозиции на блоки и построения на этих блоках независимых сеток.

Назовем множество формируемых декомпозицией блоков, описывающих исходную геометрическую модель, блочной геометрией. Блочная геометрия вместе с геометрической моделью содержат всю необходимую информацию для построения сетки. Каждый блок состоит из шести граней, грань содержит четыре ребра, а ребро — две вершины. Через введение *составных* ребер и граней реализована возможность связи двух блоков и граней по части их сторон. Более подробно такая структура описана в [3].

Блочная сетка в памяти хранится по фрагментам — с каждым блоком геометрии связана независимая структурированная сетка. Из этого представления она может быть преобразована в ячеечно-узловую или ячеечно-граневую неструктурированную сетку. Сетки на смежных блоках

совпадают *узел в узел* вследствие того, что сначала сетка строится на границах блоков.

В статье рассматриваются четыре способа, оптимизирующие построение сетки по времени и памяти:

1) параллельное построение сеток по блокам;
2) параллельное построение сетки внутри блока;
3) перестроение сеток только на измененных блоках;
4) построение сеток на блоках с их выгрузкой в файл.

Указанные способы могут применяться как совместно друг с другом, так и по отдельности. Действие реализованных алгоритмов демонстрируется на примере построения сетки для модели корпуса устройства локализации расплава [6]. Декомпозиция модели содержит 1303 блока. Количество ячеек в сетке зависит от количества узлов на ребрах блоков. Таким образом, размер сетки может варьироваться в широких пределах.

Запуск программы осуществлялся на двух различных персональный компьютерах (ПК). Первый имеет 4-ядерный процессор Intel Core i5-2400 с частотой 3,1 ГГц и доступный объем оперативной памяти 16 Гб. Второй ПК укомплектован четырьмя 6-ядерными процессорами Intel Xeon X7542 с частотой 2,67 ГГц и доступным объемом оперативной памяти 256 Гб.

1. Параллельное построение сеток по блокам

Как отмечено выше, блочная сетка в памяти хранится по фрагментам. С каждым блоком связан независимый фрагмент сетки: он может строиться на основе ранее построенных сеток на границах блоков, соседние же блоки в его построении не участвуют. Это позволяет распараллелить цикл построения сетки по блокам, если сетка на границах блоков была построена. Также реализованы информирование о ходе выполняемого процесса (с указанием выполненной части в процентах) и возможность его прерывания.

Последовательность построения блочной сетки следующая. Сначала выполняется построение одномерных сеток на ребрах, затем — поверхностных сеток на гранях и только потом — объемных сеток на блоках. Объемную сетку на блоке не удастся построить, если не были построены сетки на его гранях. Аналогично, поверхностная сетка на грани не построится, если не были построены сетки на ее ребрах.

Назовем ребро, грань и блок топологическими элементами. Сетка на топологическом элементе может быть построена независимо от сеток на других элементах того же типа. Цикл построения сеток распараллеливается с помощью средств OpenMP [7] с динамическим распределением итераций цикла между потоками. Если для топологического элемента нет сетки, то строим ее и заносим в контейнер с потокобезопасной вставкой.

Измерим время построения объемной блочной сетки с конечными размерами в 2,25, 4,58 и 8,80 млн ячеек для модели корпуса устройства локализации расплава (рис. 1, см. также цветную вкладку). Построение проводилось на описанном выше ПК с 24 процессорными ядрами. Полученные данные приведены в табл. 1.

Эффективность распараллеливания E_N и коэффициент ускорения Sp_N вычислим по формулам

$$E_N = \frac{T_1}{T_N N} \cdot 100\,\%; \quad Sp_N = \frac{T_1}{T_N},$$

где T_1, T_N — время решения одной и той же задачи на одном и N процессорах соответственно. На рис. 2, 3 (см. также цветную вкладку) представлены зависимости эффективности распараллеливания и коэффициента ускорения от количества потоков N по данным табл. 1.

Рис. 1. Фотография корпуса устройства локализации расплава, взятая из [6]

Таблица 1

Время (в с) построения сетки в зависимости от ее размера и количества потоков

Размер сетки, млн ячеек	Количество потоков										
	1	2	3	4	5	6	7	8	9	10	24
2,25	154,2	84,1	57,4	40,3	36,4	33,6	30,7	27,0	25,5	25,5	20,4
4,58	318,0	171,3	127,7	82,3	74,7	70,6	62,7	58,3	54,7	53,1	52,2
8,80	499,6	266,3	193,0	120,2	112,6	103,3	91,3	88,4	83,0	79,0	69,3

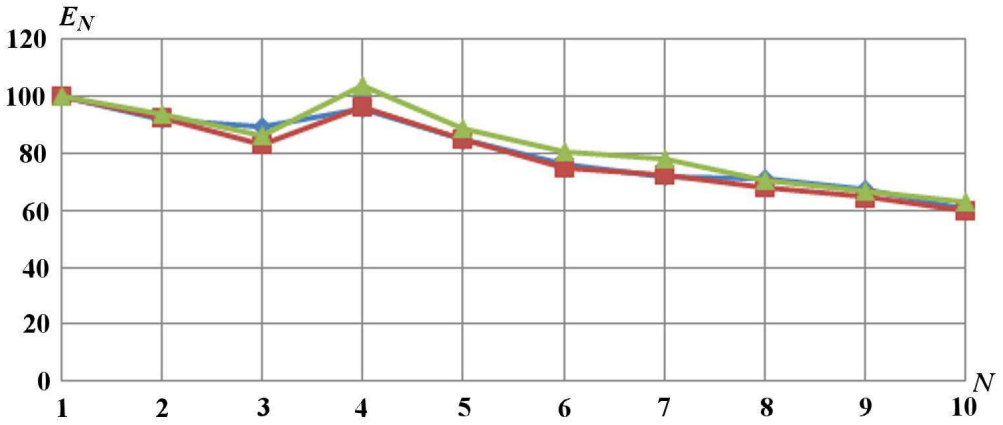


Рис. 2. Зависимости эффективности распараллеливания от количества потоков для сеток разного размера: — 2,25 млн ячеек; — 4,58 млн ячеек; — 8,80 млн ячеек

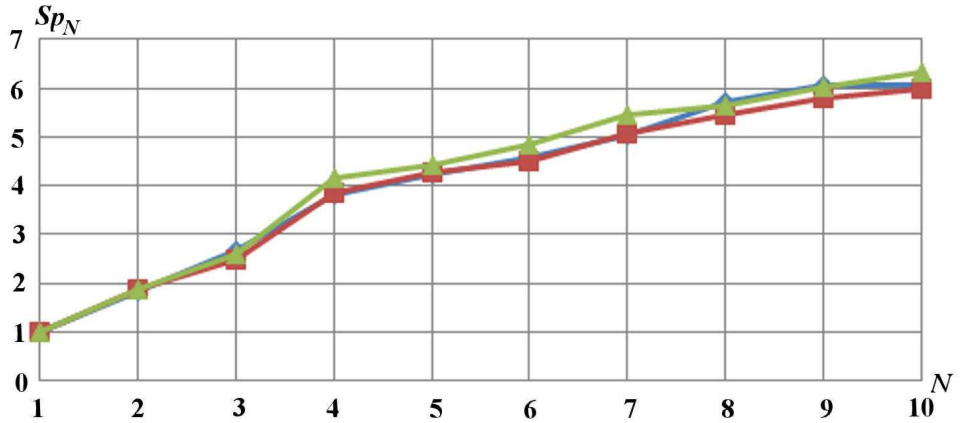


Рис. 3. Зависимости коэффициента ускорения от количества потоков для сеток разного размера: — 2,25 млн ячеек; — 4,58 млн ячеек; — 8,80 млн ячеек

На рассмотренном примере достигнуто ускорение в 6 раз с помощью 10 потоков с эффективностью 60 %. Можно и дальше увеличивать количество потоков до 24, но это дает максимальное ускорение до 7 раз с эффективностью 30 %. На обычных ПК с количеством ядер от 2 до 8 не имеет смысла задавать потоков больше, чем количество физических ядер в процессоре: в этом случае коэффициент ускорения перестает повышаться.

На рис. 2 (см. также цветную вкладку) выделяется пик эффективности распараллеливания для четырех потоков. Для него характерен больший коэффициент ускорения, чем ожидается от экстраполяции по предыдущим данным. Это можно объяснить следующим образом. Основное время при построении сетки тратится на коррекцию граничных узлов (вычисление их проекций на границу геометрической модели), которые составляют около 15 % от общего количества всех узлов (для сетки с 2,25 млн ячеек). Рассматриваемая геометрическая модель разбита на множество блоков, два из которых выделяются своими большими объемами. Это блоки, которые описывают вертикальную стенку корпуса (см. рис. 1, а также цветную вкладку). Вместе они содержат четыре большие по площади грани, на которых узлы сетки должны корректироваться. Четыре потока справляются с этим более эффективно, чем три, два или один поток.

На основе анализа этого пика можно сделать вывод, что балансировка вычислительной нагрузки встроенными средствами OpenMP не всегда эффективна. Как правило, используется дополнительная сортировка данных и их равномерное распределение по потокам. В данной работе был использован другой подход, который описан в следующем разделе.

2. Параллельное построение сетки внутри блока

Сетки на грани и блоке строятся от заданной границы. Границы задаются на основе фрагментов сеток, построенных ранее на ребрах или на гранях для случая с блоком. Этот этап назовем сборкой границы. Так как сетки на блоках структурированные, то для их представления в памяти необходимы двумерные и трехмерные массивы узлов. Внутренние узлы сетки вычисляются интерполяционным методом [8] на основе информации о граничных узлах. Чтобы распараллелить процесс вычисления внутренних узлов, выполним декомпозицию внутренней области на примерно равные по количеству узлов части (фрагменты). Опишем этот алгоритм на примере декомпозиции двумерного массива. Обобщение на трехмерный случай не вызовет затруднений.

Входными данными для алгоритма являются размеры исходного массива по двум направлениям I и J, максимальное количество фрагментов $N_{\max}$ и минимальное количество узлов $V_{\min}$ в фрагменте на выходе. На первом этапе алгоритма вычисляем количество фрагментов $N = \min\left(\max\left(\left\lfloor \frac{IJ}{V_{\min}} \right\rfloor, 1\right), N_{\max}\right)$, где $\lfloor x \rfloor$ — это наибольшее целое, меньшее или равное x. Здесь $1 \le N \le N_{\max}$. На втором этапе в цикле формируются фрагменты массива. Формирование i-го фрагмента ($i = \overline{0, N-1}$) осуществляется отбрасыванием части от оставшегося на данный момент массива (остатка) с размерами по направлениям I', J'. Для нулевого фрагмента остатком является исходный массив с размерами I, J. "Площадь" i-го фрагмента примерно равна "средней площади" фрагмента остатка: $V_{fr} = \left\lfloor \frac{I'J'}{N-i} \right\rfloor$. При этом фрагмент "отрезается" только по "длинной стороне" остатка и его размер по этому направлению равен $V_{fr}/\min(I', J')$. Часть массива, отбрасываемая при формировании i-го фрагмента, становится остатком для процедуры формирования $(i+1)$-го фрагмента.

На рис. 4 представлены три примера декомпозиции двумерных массивов. Фрагменты формировались слева направо и снизу вверх.

Минимальное количество узлов в фрагменте задается для того, чтобы ограничить декомпозицию при малых массивах. В противном случае много времени потеряется за счет накладных расходов OpenMP. Параметр $V_{\min}$ варьируется в зависимости от сложности вычисления внутреннего узла. Для текущей реализации вычислений опытным путем установлено, что $V_{\min} = 100\,000$.

После определения местоположения внутренних узлов сетки происходит их проецирование на поверхность геометрической модели. Узлы между потоками распределяются средствами OpenMP.

Для исследования времени построения сетки в зависимости от количества потоков воспользуемся блочной геометрией модели корпуса устройства локализации расплава. Построим на ней сетку, содержащую 8,80 млн ячеек, при общем количестве потоков N, кратном четырем (4, 8, 12, 16, 20, 24, 28). При этом количество одновременно обрабатываемых блоков N_b зададим равным 1, 2, 4 и 8. Соответственно внутри каждого блока может быть использовано N/N_b потоков. Полученные данные приведены в табл. 2.

Значения в табл. 2, закрашенные серым цветом, указывают на увеличение времени построения сетки из-за добавления новых потоков. Из таблицы видно, что лучший результат достига-

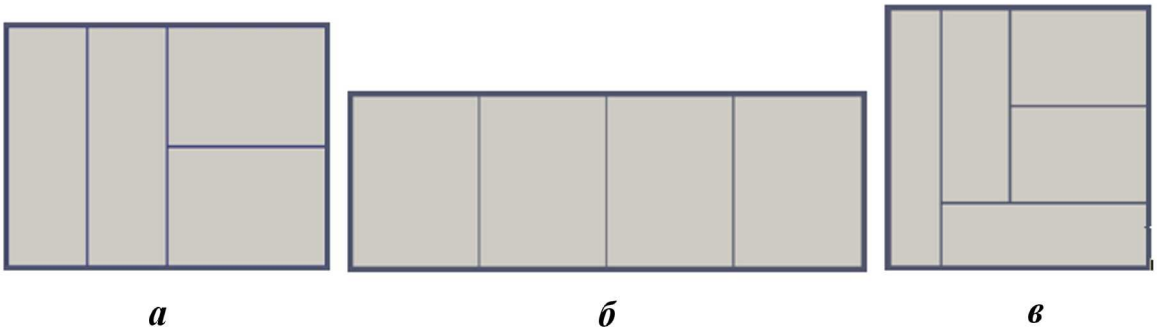


Рис. 4. Примеры декомпозиций массива: *а* — на 4 фрагмента при соотношении сторон 4×3; *б* — на 4 фрагмента при соотношении сторон 3×1; *в* — на 5 фрагментов при соотношении сторон 1×1

Таблица 2

Время (в с) построения объемной блочной сетки в зависимости от количества потоков

Кол-во одновременно обрабатываемых блоков	Общее количество потоков							
	1	4	8	12	16	20	24	28
1	544,7	148,9	97,8	84,5	71,5	70,5	75,3	74,6
2	—	142,8	81,0	66,7	55,5	51,7	48,2	47,8
4	—	123,4	74,8	64,4	50,3	41,2	40,6	38,6
8	—	—	89,7	—	53,8	—	41,1	—

ется при $N_b = 4$ для двух и более потоков внутри блока. Дополнительными подборами параметров наименьшее время построения (37,8 с) было получено на конфигурации 5 блоков по 5 потоков. Это дало ускорение в 14,4 раза с эффективностью распараллеливания 57 %. Таким образом, удалось повысить максимальное ускорение, полученное способом 1 (см. разд. 1), более чем в два раза, не уменьшив при этом эффективности.

3. Перестроение блочной сетки на измененных блоках

В ходе декомпозиции, особенно на завершающей стадии, пользовательские изменения блоков становятся локальными. В этом случае достаточно перестроить только часть сеточных фрагментов блочной сетки, что существенно ускорит получение общей результирующей сетки.

Механизм построения сетки следующий. Декомпозиция геометрии представлена связанными блоками. Каждый блок содержит топологическую и геометрическую информацию. Эта информация используется для вычисления сеточных фрагментов и объединения их в единую сетку. В ходе декомпозиции информация меняется. Появляются новые, удаляются старые блоки, некоторые из них изменяются. Изменение блока предполагает изменение связанного с ним фрагмента сетки, что приводит к его повторному перестроению. При удалении блока из декомпозиции также исключается связанный с ним сеточный фрагмент. Добавление блока приводит к добавлению нового сеточного фрагмента. Процесс приведения к согласованному состоянию блочной геометрии и сетки (синхронизация) выполняется перед каждым перестроением сетки, позволяя сохранить часть фрагментов. Основная задача, которую необходимо решить, — поиск способа, позволяющего фиксировать, накапливать и суммировать акты изменения, удаления и добавления блоков. Рассмотрим этот способ подробнее.

С каждым элементом блочной геометрии (вершиной, ребром, гранью и блоком) связано его состояние. Состояние может иметь одно из пяти значений: изменен (*mod*), добавлен (*add*), отмечен удаленным (*rem*), в очереди на удаление (*del*) и стабилен (*stab*). Состояния изменяются в зависимости от воздействий. Воздействия могут быть следующего типа: изменение (*modify*),

удаление (*delete*), сброс состояния (*reset*). Изменение состояний в зависимости от воздействий представлено на рис. 5 в виде модели конечного детерминированного автомата. Ошибочное состояние объекта отмечено как *error*. В идеальном случае перехода к нему быть не должно. Оно возникает, если пытаются удалить или изменить удаленный ранее объект, что может быть связано с явными ошибками в программе.

Каждый блок декомпозиции имеет свою копию рассмотренного выше автомата. Это позволяет процессу синхронизации иметь актуальные состояния для всех сеточных фрагментов. После ознакомления с состоянием блока посылается воздействие *reset*, что вызывает переход состояния к значению *stab* (стабилен) или *del* (в очереди на удаление).

Отметим, что предлагаемый способ может быть обобщен и на другие области применения, где есть динамически изменяющиеся объекты и стоит задача их синхронизации со связанными объектами.

Далее будет проанализировано время построения сетки для модели корпуса локализации расплава (см. рис. 1, а также цветную вкладку) при различных начальных условиях. Здесь под построением сетки подразумевается вычисление сеточных фрагментов на соответствующих топологических элементах, сборка фрагментов в единую неструктурированную сетку и ее визуализация. Все вычисления выполняются с использованием одного потока.

Построение объемной сетки с начального состояния, при котором сеточные фрагменты на топологических элементах полностью отсутствуют, занимает 153,1 с. На построение поверхностной сетки при таком же начальном состоянии уходит 143,1 с. Небольшая разница во времени связана с тем, что основная его часть тратится на проецирование граничных узлов сетки на поверхность геометрической модели. При существовании сеточных фрагментов на поверхностных гранях блоков картина иная: построение объемной сетки занимает всего 15,7 с. Как видно, достраивание объемной сетки по существующей поверхностной сетке выполняется в 10 раз быстрее.

Если же сетка строится по уже существующим сеточным фрагментам на всех присутствующих в ней элементах, то время построения сетки становится еще меньше: теперь оно складывается только из времени сборки неструктурированной сетки из сеточных фрагментов и времени ее визуализации. При таких условиях для объемной сетки оно равно 4 с, а для поверхностной 0,7 с.

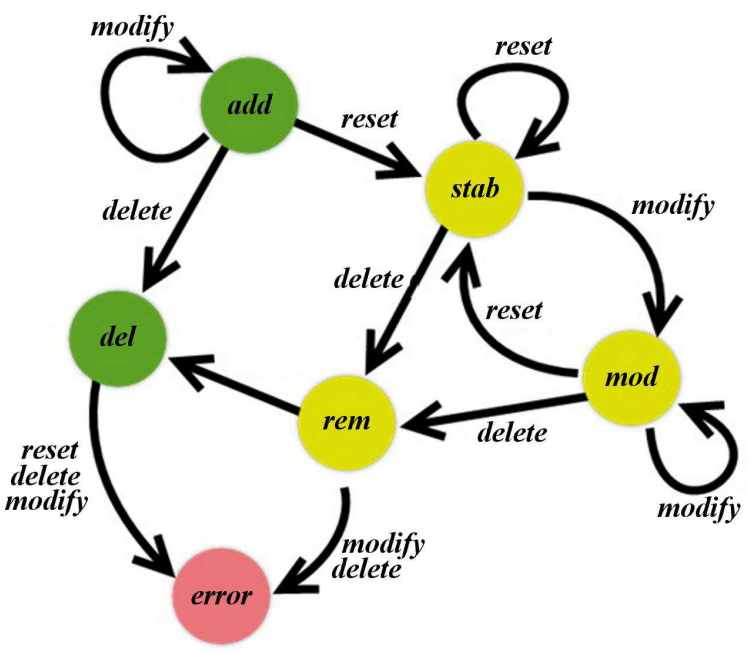


Рис. 5. Модель конечного детерминированного автомата, описывающего изменение состояния объекта

Для объемной сетки это время больше, потому что обрабатывается больший объем данных, чем в случае поверхностной сетки.

Теперь рассмотрим эффективность оптимизации в двух ситуациях — при изменении и удалении блоков. Изменение количества узлов сетки на одном ребре блока приводит к необходимости задания такого же их количества на всех параллельных ребрах связанных блоков. В противном случае структурированные сетки на гранях и блоках не могут быть построены. Изменения сетки на ребре приводят к изменению сеток на связанных с ним гранях и блоках. В рассматриваемом примере после изменения количества узлов на одном ребре объемная сетка была построена за 8 с. Вторая ситуация — это удаление блоков, описывающих опоры устройства локализации расплава. После таких изменений объемная сетка была построена за 2,1 с. Удаление блоков привело к исключению сеточных фрагментов и уменьшению общего размера сетки. Это, в свою очередь, уменьшило время сборки сетки и ее визуализации. Полученное время в двух рассмотренных ситуациях является относительным и зависит от объема затронутых изменений, но в любом случае оно не превышает времени построения сетки при полном отсутствии сеточных фрагментов.

Прерывание построения сетки в середине процесса пользователем или из-за ошибки не приводит к потере промежуточных результатов. После повторного запуска работа начинается с прерванного места.

4. Пофрагментное построение сетки с выгрузкой в файл

Уменьшение времени построения сетки не в полной мере решает проблему построения больших сеток. Для этого необходимо оптимизировать использование оперативной памяти. Выгрузка фрагмента построенной сетки в файл позволяет строить на ПК сетки, не помещающиеся целиком в оперативную память. Например, такие сетки характерны для моделей тепловыделяющих сборок (ТВС) атомных реакторов, в которых количество ячеек доходит до 1 млрд. Для подобных случаев разработан алгоритм записи сетки в файлы по фрагментам. В препроцессоре пакета программ ЛОГОС запись для объемных сеток реализована в распределенный файл формата ЕФР [9, 10].

В общих чертах алгоритм, оптимизирующий использование оперативной памяти, выглядит следующим образом. Для каждого блока декомпозиции строится граничная сетка. На основе граничной сетки строится сетка на блоке. Построенная сетка записывается в файл и удаляется из оперативной памяти. Граничная сетка также удаляется, если на остальных смежных с ней блоках сетка уже построена. Таким образом, потребление оперативной памяти не превысит ее объема для хранения самого большого фрагмента сетки с учетом построенных, но не удаленных граничных сеток.

Запуск программы, реализующей данный способ, выполнялся на вышеописанном ПК с 4-ядерным процессором. В препроцессор были загружены исходная геометрическая модель и ее блочное представление для корпуса устройства локализации расплава (см. рис. 1, а также цветную вкладку). В оперативной памяти все это занимало 350 Мб. Построение сетки выполнялось в двух однопоточных режимах: 1) с формированием сетки в оперативной памяти; 2) с записью в файл по фрагментам. Активность использования ресурсов анализировалась с помощью программы Process Explorer [11].

На рис. 6 представлены иллюстрации активности использования процессора, оперативной и дисковой памяти. Для первого режима характерны полная загрузка ядра процессора, отсутствие взаимодействия с диском и увеличивающееся потребление оперативной памяти. Участок с быстрым ростом потребления памяти связан с этапом построения объемных сеток и выделением памяти под объединенную сетку. Дальнейший горизонтальный участок связан с наполнением объединенной сетки узлами и ячейками и с визуализацией. Объем потребляемой оперативной памяти по сравнению с исходным увеличился на 650 Мб. Для режима с записью в файл по фрагментам характерны неполная и изменяющаяся во времени загрузка ядра процессора, активный вывод данных на диск и относительно постоянный объем используемой оперативной памяти на всем протяжении времени построения сетки.

Разработанный алгоритм был также применен для построения сетки на модели ТВС (рис. 7).

На рис. 8 представлена использованная в ТВС декомпозиция геометрии ТВЭЛа на блоки и связанная с ней демонстрационная блочная сетка небольшого размера. В действительности сетка состояла из 1 млрд ячеек и узлов и 3,2 млрд

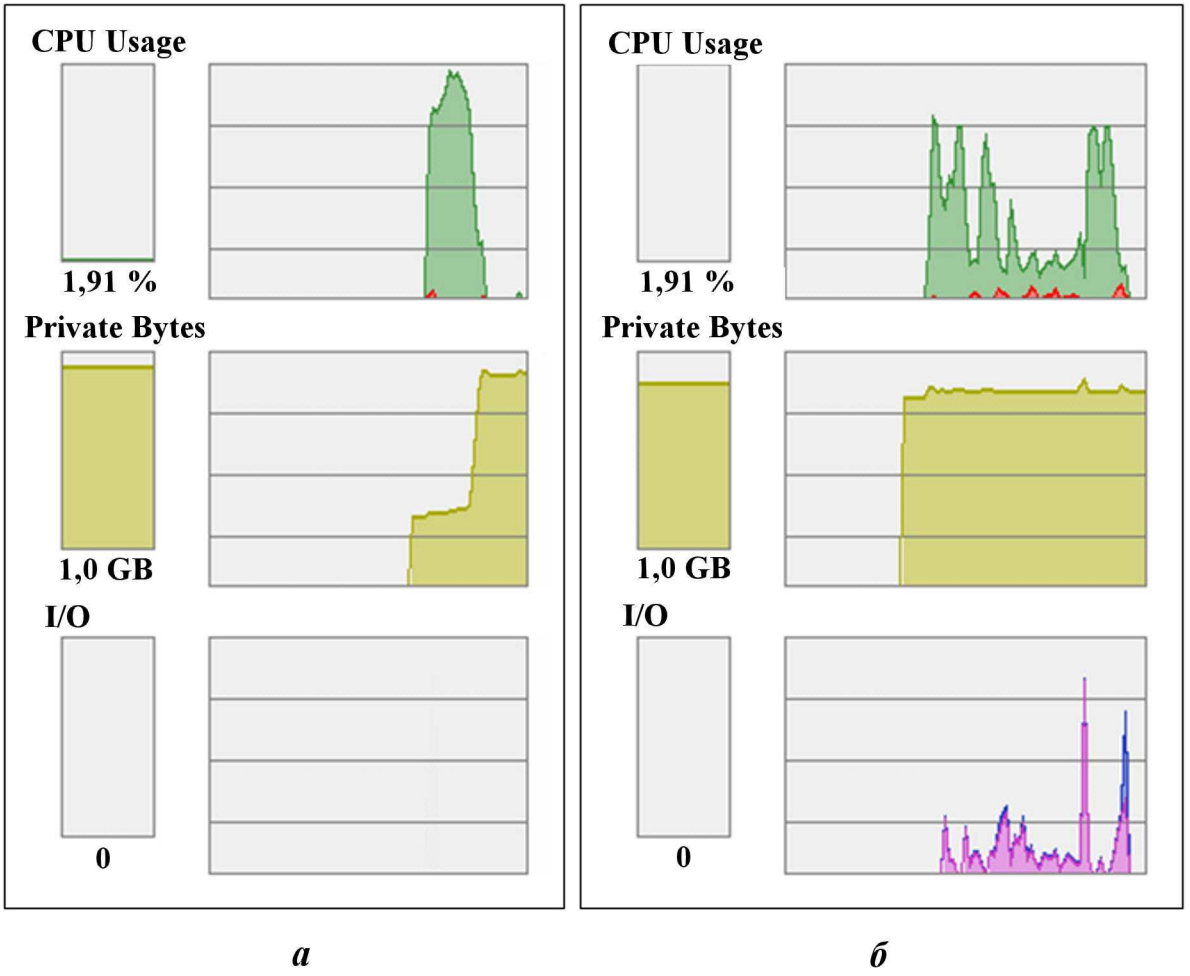


Рис. 6. Снимки экрана Process Explorer при измерении активности процесса построения сетки с записью в ОЗУ (*а*) и в файл (*б*)

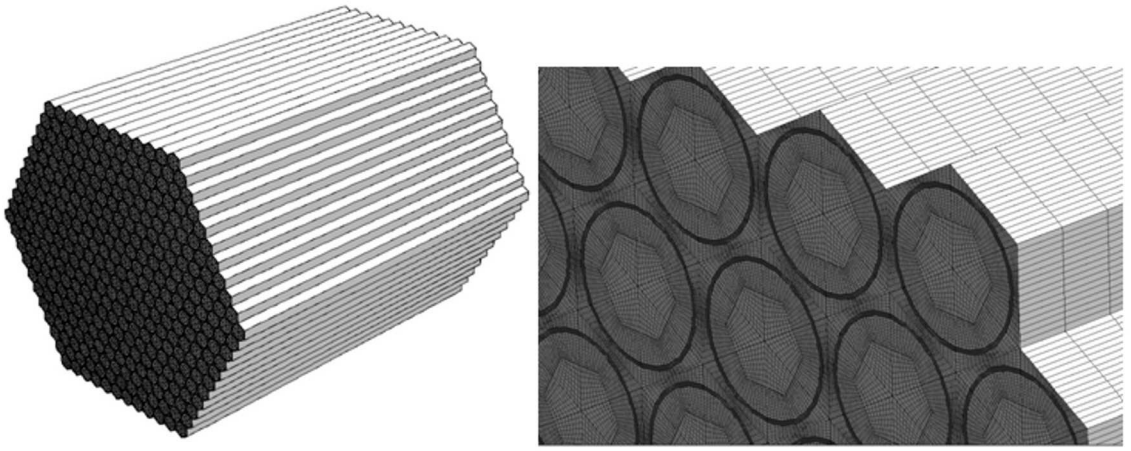
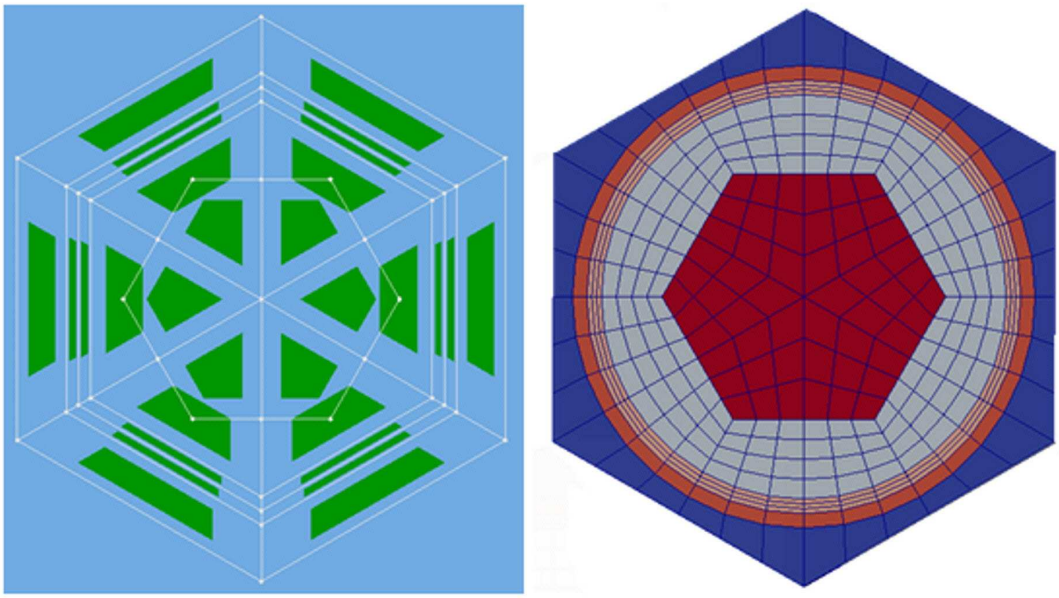


Рис. 7. Сетка на ТВС из 1 млрд ячеек



а *б*

Рис. 8. Декомпозиция геометрии ТВЭЛа на блоки (*а*) и блочная сетка (*б*)

граней. Данные были сохранены в 8 500 фрагментах общим размером занимаемого дискового пространства 250 Гб. Построение сетки заняло 4 часа. Пиковое потребление памяти составило 1 Гб.

Заключение

Предложенные четыре способа, оптимизирующие построение сетки, могут применяться как отдельно, так и в совокупности.

Совместное использование поблочного и внутриблочного распараллеливания с перестроением сеток на измененных блоках уменьшает время построение сетки до нескольких секунд вместо нескольких минут. Для случаев, когда необходимо часто перестраивать сетку, время выигрыша является существенным.

Использование пофрагментного построения сетки с выгрузкой в файл позволяет строить на ПК сетки, состоящие из 1 млрд и более ячеек. При этом максимальная загрузка оперативной памяти не превышает ее объема для хранения наибольшего сеточного фрагмента в декомпозиции геометрии.

Предложенные алгоритмы реализованы и внедрены в препроцессор ЛОГОС, начиная с версии 5.1.

Список литературы

1. *Лазарев В. В., Фархутдинов В. Ф., Данилова Т. Г., Морозова Е. В.* Блочные регулярные сетки в препроцессоре ЛОГОС // Молодежь в науке: сб. докл. 12-й науч.-тех. конф. Саров: ФГУП "РФЯЦ-ВНИИЭФ", 2014. С. 67—70.
2. *Фархутдинов В. Ф., Тарасов В. И., Соловьев А. Н., Борисенко О. Н., Лазарев В. В., Логинов Д. В., Аверина Н. С., Марунин А. В., Гриднев А. И., Шабунина М. Г., Кузнецов М. Г., Черенкова М. В., Фролова Е. А., Лукичев А. Н., Смолкина Д. Н., Купалова А. Г., Кузьменко М. В., Ховрин Н. А.* Обзор возможностей по обработке геометрий и построению сеточных моделей в ЛОГОС.ПРЕПОСТ // Супервычисления и математическое моделирование. Тр. XIII Межд. семинара / Под ред. Р. М. Шагалиева. Саров: ФГУП "РФЯЦ-ВНИИЭФ", 2012. С. 524—533.
3. *Лазарев В. В., Борисенко О. Н., Фархутдинов В. Ф., Васильев Г. А., Мартенс Р. В.* Декомпозиция геометрии на шестигранные и четырехугольные блоки для построения расчетных сеток // Молодежь в науке: сб.

докл. 14-й науч.-тех. конф. Саров: ФГУП "РФЯЦ-ВНИИЭФ", 2015. С. 57—60.

4. Parallel Part Meshing in ANSYS Mechanical: The Need for Speed. http://www.ansys-blog.com/parallel-part-meshing-in-ansys-mechanical.
5. Модуль Pamgen проекта Trilinos. https://trilinos.org/packages/pamgen.
6. Российское атомное сообщество. "Атоммаш" изготовит усовершенствованную "ловушку" расплава для БалтАЭС. http://atomic-energy.ru/news/2012/03/07/3163/
7. OpenMP Architecture Review Board. www.openmp.org
8. *Thompson J. F.* Handbook of Grid Generation / Ed. by J. F. Thompson, B. K. Soni, N. P. Weatherill. New York: CRC Press, 1999.
9. *Олесницкая К. К., Антипин И. А., Шубина М. А.* Библиотека ЕФР для масштабируемого доступа к файловым данным на многопроцессорных ЭВМ // Супервычисления и математическое моделирование. Тр. XIII Межд. семинара / Под ред. Р. М. Шагалиева. Саров: ФГУП "РФЯЦ-ВНИИЭФ", 2012. С. 370—379.
10. *Олесницкая К. К., Антипин И. А., Петрова М. А.* Библиотека ЕФР как средство эффективного доступа к файловым данным на гибридных вычислительных системах и суперкомпьютерах // Супервычисления и математическое моделирование. Тр. XV Межд. конф. / Под ред. Р. М. Шагалиева. Саров: ФГУП "РФЯЦ-ВНИИЭФ", 2015. С. 346—354.
11. Process Explorer Official Webpage. https://technet.microsoft.com/en-us/sysinternals/bb896653.

Статья поступила в редакцию 04.08.17.

PARALLELIZATION AND OPTIMIZATION OF THE CONSTRUCTION OF BLOCK COMPUTATIONAL GRIDS IN PREPROCESSOR OF "LOGOS" PROGRAM PACKAGE / V.V. Lazarev (FSUE "RFNC-VNIIEF", Sarov, N. Novgorod region).

The methods optimizing the block computational grid construction using the previously made decomposition of a geometric model into blocks are considered. Data structures representing the decomposition and block grids are described. The parallelized algorithms of constructing a block grid are described on the base of these representations. The plots of the parallelization efficiency and speedup versus the number of threads are presented. An algorithm that allows reconstructing a grid on the varied blocks only with respect to the modified decomposition into blocks has been also developed. For grids of sizes beyond the memory size, the fragment-by-fragment construction method with recording to separate files is proposed. A grid of 1 billion cells has been constructed. The developed algorithms have been implemented in the LOGOS preprocessor beginning from version 5.1.

Keywords: parallelization, OpenMP, a block computational grid, geometry decomposition into blocks, preprocessor of the LOGOS program package.