АЛГОРИТМЫ ГЕОМЕТРИЧЕСКО-ТОПОЛОГИЧЕСКОЙ ДЕКОМПОЗИЦИИ НЕСТРУКТУРИРОВАННОЙ СЕТКИ ДЛЯ СЛУЧАЯ РАСПРЕДЕЛЕННОГО ПРЕДСТАВЛЕНИЯ ДАННЫХ, РЕАЛИЗОВАННЫЕ В МЕТОДИКЕ ТИМ

Ю. Н. Кащеев, И. Г. Новиков, А. А. Пушкарёв

ФГУП «РФЯЦ-ВНИИЭФ», г. Саров Нижегородской обл.

В докладе приводится описание алгоритмов геометрическо-топологической декомпозиции, реализованных в методике ТИМ [1], с использованием графа на основе распределенного представления математических областей. В методике ТИМ счет ведется в многообластном приближении, каждая деталь конструкции – отдельная математическая область. Чтение информации о математической области производится в память одного вычислительного узла. В связи с этим встает вопрос о максимально возможном размере математической области. Представление математической области в распределенном между всеми процессами виде снимает ограничение на объем оперативной памяти вычислительного устройства при чтении информации и позволяет использовать сетки порядка нескольких сотен миллионов ячеек в одной математической области.

Алгоритм распределенной геометрическо-топологической декомпозиции состоит из двух этапов: геометрического и топологического. На первом этапе отбираются все ячейки, центры которых находятся в пределах ограничивающей сферы. На втором этапе выполняется проверка топологической связности списка ячеек.

Представление информации о математических областях

Чтение из файла информации о математических областях в методике ТИМ является двухпроходным. При первом проходе управляющим MPI-процессом зачитывается минимальная информация о математической области, необходимая для выполнения декомпозиции. Этот этап называется чтение «минимальной математической области», при котором зачитывается топология и координаты узлов сетки, необходимой для геометрическо-топологической декомпозиции. После декомпозиции, на основе этой полученной информации выполняется формирование параобластей. Далее выполняется второй проход чтения, когда управляющий процесс математической области зачитывает счетные величины и осуществляется их распределение в соответствующие параобласти. Такой подход позволил проводить трехмерные расчеты до 100 млн. ячеек в одной математической области. При зачитывании данных из математических областей, число ячеек в которых превышает 100 млн. ячеек, происходило переполнение памяти, доступной на одном вычислительном узле. Для снятия ограничений на количество ячеек в математической области необходимо проводить чтение информации в распределенном между MPI-процессами режиме. Для этого были разработаны алгоритмы чтения файла-разреза на всех этапах в полностью распределенном виде. В этом случае чтение данных математических областей осуществляется не одним, а всеми MPI-процессами. Распределение между MPI-процессами делается по номерам элементов сетки. При этом весь набор элементов сетки распределяется между выделенными MPI-процессами. Для равномерного использования оперативной памяти процессов, количество элементов сетки между ними распределяется максимально равномерно. Например, если математическая область состоит из 100 ячеек и в ее расчете участвуют 5 процессов в рамках одинаковых по производительности вычислительных устройств, то распределение ячеек между ними будет следующим:

- на нулевом с 1 по 20;
- на первом с 21 по 40;
- на втором с 41 по 60;
- на нулевом с 61 по 80;
- на нулевом с 81 по 100.

Данный подход позволяет полностью снять ограничение на количество ячеек в одной математической области, связанное с нехваткой оперативной памяти на узлах.

Однако, существуют и недостаток, в виде увеличения времени поиска информации о топологическом соседстве элементов сетки, по причине выполнения дополнительных MPI-обменов.

Геометрическо-топологическая декомпозиция

Геометрическо-топологическая декомпозиция основана на топологическом соседстве ячеек между собой. Помимо соседства ячеек сетки используется еще и геометрический критерий, ограничивающий попадание ячеек в параобласть при их наборе. При построении декомпозиции геометрическо-топологическим алгоритмом гарантирована односвязанность параобластей, относительная гладкость границ.

Декомпозиция области проводится с учетом топологического соседства ячеек с использованием координат центров ячеек и координат узлов сетки таким образом, чтобы число ячеек в каждой параоб-

ласти было примерно одинаковым, либо сбалансированным по вычислительной нагрузке.

Алгоритм разбиения области на параобласти строится следующим образом. Зная количество ячеек сетки N, можно получить количество ячеек *NPar* в каждой параобласти $p = 1, nP$:

$NPar(p) = \frac{N}{nP} + 1$, для первых m параобластей, где m – остаток деления $\frac{N}{nP}$, nP – итоговое количество параобластей, $NPar(p) = \frac{N}{nP}$, для остальных $nP - m$ параобластей.

Определяются Xmin и Xmax, Ymin и Ymax, Zmin и Zmax, – глобальные максимумы и минимумы координат узлов исходной области. Находится ближайший к вершине параллелепипеда узел области {Xmin, Ymax, Zmax}. Этот узел становится начальным (опорным) в разбиении.

Затем определяется стартовая ячейка, одной из вершин которой является начальный (опорный) узел, и ее центр r_0. Для этой ячейки определяется максимальная диагональ (расстояние между самыми удаленными друг от друга вершинами ячейки). Вычисляется некоторый максимальный параметр ΔR, с которым будет увеличиваться радиус сферы, в пределах которого будут набираться ячейки в параобласть итерационно. Этот параметр выбран эмпирически и составляет четыре-пять максимальных диагоналей стартовой ячейки. После чего, по топологическому соседству ячеек, начиная от стартовой ячейки, производится набор ячеек-претендентов в параобласть, которые попадают в сферу радиуса $R = \Delta R \cdot i : \left|\vec{r}_k - \vec{r}_0\right| < R$, где $\vec{r}_k$ – центр k-той ячейки-претендента, $i = 1,...$ – номер итерации увеличения радиуса при наборе параобласти.

Если центр ячейки находится в пределах заданной сферы, то ячейке k ставится признак принадлежности к параобласти p. И счетчик количества ячеек K увеличивается на единицу. Если больше нет ячеек, попадающих в сферу радиуса R, а количество ячеек $K < NPar(p)$, тогда переходим к следующей итерации по увеличению радиуса сферы: $i = i + 1$. Количество итераций по увеличению радиуса ограничено пятью пустыми итерациями подряд для экономии времени. Под пустой итерацией понимается итерация, на которой не произошло добавление ячеек в рассматриваемую параобласть.

Завершается набор ячеек в параобласть p тогда, когда будет выполнено условие $K \geq NPar(p)$ или выполнены все итерации по увеличению радиуса.

Данный подход к разбиению области на параобласти не обладает недостатками чисто геометрического подхода и является наиболее подходящим для декомпозиции данных по процессам в методике ТИМ.

После завершения построения очередного компакта формируются его граничные признаки, а также определяются угловые ячейки между компактами. Из списка угловых ячеек выбирается первая, среди соседей которой есть ячейка, не входящая ни в один компакт. Данная ячейка будет стартовой (опорной) при построении нового компакта.

Алгоритм построения компактов будет выполняться до тех пор, пока всем ячейкам математической области не будет выставлен признак принадлежности компактам.

Из-за разномасштабности формы ячеек и сильного отличия их «весов» количество сформированных компактов может быть больше запланированного. Для приведения количества компактов к запланированному значению, выполняется операция объединения компактов. Для этого компакты перенумеровываются в порядке возрастания «весов», и для компакта с наименьшим «весом» ищется соседний наименьший компакт. После чего эти компакты объединяются в один. Операция объединения выполняется до тех пор, пока количество компактов не будет приведено к изначальному разбиению.

Операция объединения допускает возможность появления компактов, «веса» которых значительно превышают допустимый «вес». Это приведет к разбалансировке по вычислительной нагрузке. Если дисбаланс составляет более 10 %, в этом случае выбирается другая угловая ячейка математической области и выполняется повторное формирование компактов. Итерации по выбору наиболее оптимальной декомпозиции выполняются до тех пор, пока разбалансированность между компактами не составит менее 10 %. Количество итераций ограничено количеством угловых ячеек математической области. В случае если все итерации построения декомпозиции завершились, то для получения итоговой декомпозиции выбирается угловая ячейка, при которой разбалансированность вычислительной нагрузки минимальна.

На рис. 1 продемонстрирован результат работы геометрическо-топологической декомпозиции в двумерном случае.

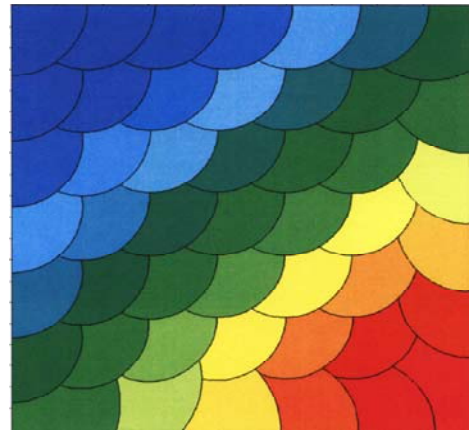

Рис. 1. Результат работы геометрическо-топологической декомпозиции

Геометрическо-топологическая декомпозиция в распределенном виде

В случае распределенного представления математических областей, каждый MPI-процесс хранит информацию только о диапазоне номеров элементов сетки. В методике ТИМ в качестве формата хранения топологии сетки используется формат по граням, который имеет следующий вид:

– для ячейки хранится неупорядоченный список всех ее граней;

– для грани хранятся номера двух формирующих эту грань ячеек; для грани, которая лежит вдоль границы, первым номером является номер граничного условия. Также для грани хранится список узлов ее формирующих, причем со стороны первой, формирующей грань ячейки узлы упорядочены против часовой стрелки, а со стороны второй – по часовой стрелке (рис. 2);

– для узла хранится номер одной из граней, сходящихся в данном узле. Для граничного узла хранится номер одной из граничных граней.

Связь между элементами сетки можно представить следующим образом:

ячейка ↔ грани → узлы → грань.

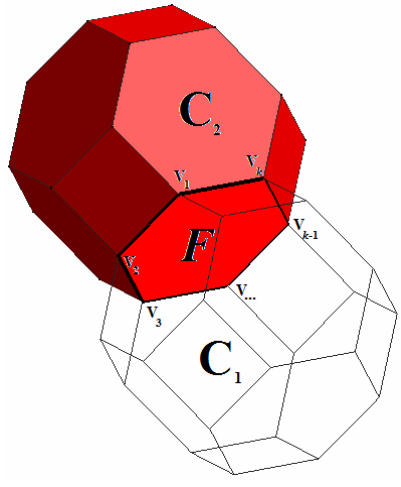

Рис. 2. Обход узлов грани относительно формирующих ее ячеек

Таким образом, получение соседних ячеек для данной ячейки идет через топологию граней и необходим MPI – обмен между процессами, так как информация о гранях может храниться на других процессах.

В алгоритмах геометрическо-топологической декомпозиции одной из наиболее часто используемых операций является поиск соседних ячеек для ячейки. Получение этой информации выполняется из распределенного графа.

Граф устроен следующим образом: вершины графа соответствуют ячейкам сетки, а ребра – соседству между ячейками. Учитывается только соседство через грань (в двумерном случае – ребро) и граничные условия.

Одно из главных отличий в работе блока декомпозиции для распределенного случая от блока декомпозиции нераспределенного случая – использование всех процессов задачи для каждой математической области. Если для нераспределенного случая декомпозиция выполнялась на управляющем процессе математической области, то для распределенного случая участвуют все процессы задачи.

На первом этапе алгоритма распределенной геометрическо-топологической декомпозиции выполняется первоначальное разбиение математической области на компакты «по весам». Каждый MPI-процесс из своего диапазона элементов вычисляет глобальные максимумы и минимумы математической области (XMin и XMax, YMin и YMax, ZMin и ZMax). Далее формируется единый список угловых ячеек математической области. В качестве стартовой ячейки, от которой начнется формирование первого компакта, выбирается угловая ячейка, центр которой максимально близок к точке с координатами {XMin, YMax, ZMax}. Список угловых ячеек формируется следующим образом: сначала каждый MPI-процесс формирует его локально из диапазона принадлежащих ему ячеек, затем локальные списки отсылаются на управляющий MPI-процесс математической области, на котором формируется единый список угловых ячеек области. Управляющий процесс математической области рассылает этот список остальным процессам с помощью функции широковещания. После чего на каждом процессе определяется стартовая ячейка.

Алгоритм распределенной геометрическо-топологической декомпозиции является итерационным, т. е. формирование компакта происходит в несколько итераций, пока не достигнет необходимого «веса». Отбор ячеек в компакт осуществляется в рамках сферы с заранее определенным радиусом, значение которого увеличивается при каждой итерации. Однако необходимо отметить, что время выполнения декомпозиции данных математической области зависит от количества выполненных итераций, поэтому желательно определить такой радиус, при котором будет выполнено минимальное количество итераций. В данной работе значение радиуса сферы определяется по формуле 1, которая подобрана эмпирически:

$$r = \frac{L}{\sqrt{nP}} \qquad (1),$$

где L – расстояние между точками, с координатами {XMax, YMax, ZMax} и {XMix, YMin, ZMin}, а nP – количество исходных компактов.

Опишем работу алгоритма. Вначале каждый процесс «проходит» по своему диапазону элементов и отбирает среди них не промаркированные (не принадлежащие ни одному компакту) ячейки, которые находятся в пределах радиуса определяемой сферы (геометрический этап). После формирования локального списка ячеек-претендентов в набираемый компакт, каждый процесс отсылает его на управляющий процесс математической области, на котором и про-

исходит формирование общего списка ячеек-претендентов. После сбора локальных списков в единый список, при первой итерации в него добавляется стартовая ячейка (при последующих итерациях, кроме стартовой ячейки добавляются и все промаркированные ячейки текущего компакта). Общий список в дальнейшем будет использоваться всеми процессами, участвующими в декомпозиции данных математической области, для этого применяется функция библиотеки MPI. Для исключения возможности набора несвязных компактов, выполняется топологический этап, который заключается в отборе не промаркированных ячеек, имеющих единую группу связности со стартовой ячейкой. Формирование списка групп связности для ячеек-претендентов выполняется на управляющем процессе, который рассылается всем, участвующим в декомпозиции области MPI-процессам. Затем, при помощи алгоритма быстрого поиска, ищется в общем списке стартовая (исходная) ячейка компакта и определяется ее группа связности. Каждый MPI-процесс отбирает из общего списка ячеек-претендентов те ячейки, которые принадлежат его диапазону, а также находятся в одной группе связности со стартовой ячейкой. Для отобранных ячеек на каждом MPI-процессе формируются два дополнительных массива. В первом хранится расстояние от центра стартовой ячейки компакта до центра каждой ячейки. Во втором – «вес» каждой ячейки. После формирования необходимых массивов, на управляющем процессе математической области выполняется сравнение суммы «весов» отобранных связных ячеек с теоретическим «весом» компакта. В случае, если сумма «весов» ячеек не превышает заданный «вес» компакта или это превышение незначительно, в пределах допустимого отклонения, не промаркированные ячейки добавляются в компакт. Если «вес» формируемого компакта меньше его допустимого «веса», то выполняется следующая итерация, с последовательностью действий, описанных выше. При каждой новой итерации радиус сферы, в пределах которой строится компакт, увеличивается в 1,5 раза.

Когда сумма «весов» отобранных по связности ячеек превышает допустимый предельный «вес» компакта, выполняется сортировка этих ячеек в порядке увеличения расстояния от их центров до центра стартовой ячейки. Из отсортированного списка ячеек отбираются ячейки, сумма «весов» которых удовлетворяет предельному «весу» формируемого компакта. После чего выполняется проверка топологической связности.

Алгоритм поиска групп связности

Одним из важнейших этапов в работе распределенной декомпозиции является поиск групп связности для распределенного списка ячеек. Так как на практике размерность списка ячеек, для которого необходимо найти связные группы, не очень большая относительно общего числа ячеек (не более десяти миллионов в задачах с общим числом в математической области порядка нескольких сотен миллионов ячеек), то используется следующий подход. Сначала по распределенному графу области строится на управляющем процессе нераспределенный граф, в котором участвуют только ячейки из заданного списка. Далее, для построенного графа, выполняется алгоритм Сомана [2] поиска связных компонент графа.

Алгоритм основан на распространении меток по соседям и в общем виде описывается следующим образом. На первом этапе всем вершинам графа проставляется метка, равная номеру данной вершины. Далее, итерационно, для каждой вершины ставится метка, равная наименьшей метке из соседей данной вершины или самой этой вершины. Таким образом, через определенное количество итераций, граф разобьется по меткам на один или несколько звездообразных подграфов, форма которого представлена на рис. 3. Количество таких графов и есть число групп связности.

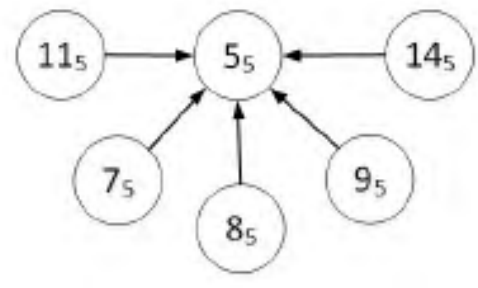

Рис. 3. Форма связной компоненты графа

В алгоритме Сомана используются итерации двух видов: по ребрам и узлам. Узлы ребра должны принадлежать одной связной компоненте. Поэтому в цикле по ребрам проверяется, одинаковые ли метки имеют узлы ребра. Если метки разные, то для узла с большей меткой ставится меньшая метка. После работы цикла по ребрам получается картина, изображенная на рис. 4.

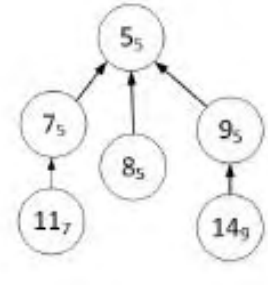

Рис. 4. Граф, получающийся после работы цикла по ребрам

Далее, в цикле по узлам для каждого узла рекурсивно вычисляется его метка по правилу Label(u) = Label(Label(u)). После этого граф примет вид, изображенный на рис. 3 и будет состоять из отдельных подграфов, являющихся связными компонентами.

Алгоритм построения нераспределенного графа из распределенного и алгоритм Сомана поиска групп связности графа распараллелены на общей памяти с применением интерфейса OpenMP.

Сортировка

В процессе работы алгоритмов геометрическо-топологической декомпозиции достаточно часто используется операция поиска элемента в массиве. Для использования быстрого поиска необходимо, чтобы список был отсортирован. Ранее в методике ТИМ использовалась быстрая сортировка с выбором центрального элемента. Для ускорения выполнения процедуры сортировки была проведена ее модификация, заключающаяся в следующем:

- ✓ При малом числе элементов использование сортировки вставками;
- ✓ изменение алгоритма для выбора сравниваемого значения;
- ✓ OpenMP-распараллеливание с использованием механизма подзадач.

Хотя число операций быстрой сортировки составляет в общем случае $O(n\log n)$, а для сортировки вставками оно составляет $O(n^2)$, на практике для массивов малой размерности выгоднее использовать именно сортировку вставками из-за меньшей константы при множителе порядка. Поэтому на последних уровнях рекурсии, когда число элементов в сортируемом массиве не превышает 20, используется сортировка вставками.

Для более равномерного разбиения массива на две части с целью уменьшения числа рекурсивных вызовов был модифицирован алгоритм определения значения, с которым сравниваются элементы массива. Был реализован динамический выбор сравниваемого значения, описанный в работе [3]. На первом вызове в качестве сравниваемого значения берется значение последнего элемента, а на всех последующих рекурсивных вызовах подсчитываются четыре числа – количество и сумма элементов, меньших сравниваемого значения, количество и сумма элементов, больших или равных сравниваемому значению: CountLess, SumLess и CountLarger, SumLarger. И в качестве сравниваемого значения для левого массива берется $\frac{\text{SumLess}}{\text{CountLess}}$, а для правого массива $\frac{\text{SumLarger}}{\text{CountLarger}}$. Такая процедура с динамическим выбором сравниваемого значения позволяет последовательно разбивать списки на приблизительно равные по количеству элементов части.

Быстрая сортировка относится к иррегулярным приложениям, для которых не существует на сегодняшний день шаблона распараллеливания с хорошей эффективностью на общей памяти для многоядерных процессоров. Это связано с тем, что в таких приложениях, как правило, отсутствуют простые циклы, которые могут быть легко распараллелены с помощью интерфейса OpenMP. К таким приложениям относятся, например, обработка связанного списка, в котором следующий элемент определяется через указатель в текущем элементе, множество алгоритмов на графах и деревьях, где требуется проход от вершины к дочерним вершинам или листьям. Чаще всего для таких алгоритмов характерен принцип «разделяй и властвуй» («*divide and conquer*»), в котором задача бьется на подзадачи, находятся решения каждой из подзадач и сливаются в одно решение. При этом для подзадач обычно происходит рекурсивный вызов, пока подзадача не окажется простой для ее прямого решения.

В стандарте OpenMP, начиная с версии 3.0, появился механизм для работы с вложенными подзадачами, который позволяет выполнять данные действия параллельно несколькими нитями.

Формат вызова в общем случае такой:

```
!$OMP TASK
 Подзадача 1
!$OMP END TASK
.....
!$OMP TASK
 Подзадача N
!$OMP END TASK
!$OMP TASKWAIT
```

Каждая подзадача, выполняемая одной нитью, заключается между директивами !$OMP TASK и !$OMP END TASK. Если для общего решения задачи необходимо, чтобы все подзадачи были выполнены, добавляется директива синхронизации для подзадач !$OMP TASKWAIT (аналог !$OMP BARRIER для нитей). На примере псевдокода быстрой сортировки покажем принцип работы данного механизма.

```
!$OMP PARALLEL
!$OMP SINGLE
call QuickSort(A, 1, N)
!$OMP END SINGLE
!$OMP END PARALLEL

subroutine QuickSort(1, p, r)
 if (r-p <= 1) then
  return
 else
  q = partition(A, p, r)
 endif
 !$OMP TASK SHARED(A)
 call QuickSort(A,p,q-1)
 !$OMP END TASK
 !$OMP TASK SHARED(A)
 call QuickSort(A,q,r)
 !$OMP END TASK
 !$OMP TASKWAIT
end subroutine QuickSort
```

Алгоритмы сортировки, используемые в геометрическо-топологической декомпозиции, были распараллелены с использованием описанного выше подхода на общей памяти с использованием интерфейса OpenMP.

Примеры декомпозиции

В качестве первого примера выбрана многообластная задача, суммарное число ячеек в которой ~307 млн. Запуск задачи выполнен на 40 вычислительных узлах (в конфигурации 1 MPI × 10 OpenMP). Время выполнения декомпозиции составило 183 сек.

В качестве второго примера выбрана однообластная задача, суммарное число ячеек в которой ~330 млн. Запуск задачи выполнен на 50 вычислительных узлах (в конфигурации 1 MPI × 10 OpenMP). Время выполнения декомпозиции составило 528 сек.

Результаты декомпозиции представлены на рис. 5–6.

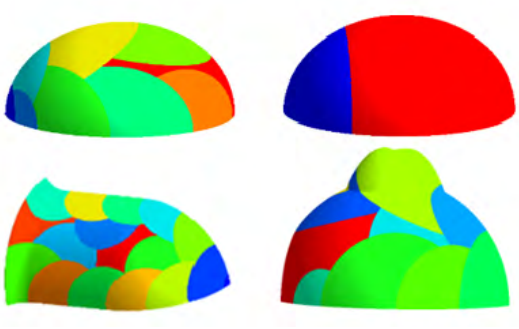

Рис. 5. Декомпозиция многобластной задачи, различные математические области разбиты на компакты

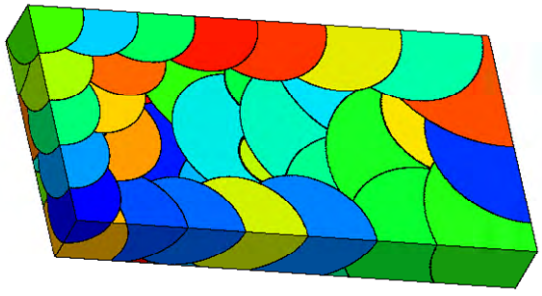

Рис. 6. Декомпозиция однообластной задачи

Заключение

Представленный в данной работе модифицированный геометрическо-топологический алгоритм позволяет выполнять декомпозицию математических областей порядка нескольких сотен миллионов ячеек за приемлемое время в полностью распределенном режиме.

Это позволило, для методики ТИМ, снять ограничение на максимально возможное количество точек в одной математической области, связанное с доступной в рамках одного узла вычислительной системы оперативной памяти. Данные алгоритмы могут быть использованы и для других методик, базирующихся на неструктурированных сетках, где вопросы декомпозиции данных по пространству стоят довольно остро.

Алгоритм распараллелен с использованием функций библиотеки MPI и с использованием интерфейса OpenMP**.**

Литература

1. Соколов С. С., Панов А. И., Воропинов А. А. и др. Методика ТИМ расчета трехмерных задач механики сплошных сред на неструктурированных многогранных лагранжевых сетках // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 2005. Вып. 3. С. 37–52.

2. Soman, Kishore and Narayanan. A fast GPU algorithm for graph connectivity, 2010.

3. Abdel latif Abu Dalhoum, Thaer Kobbay, Azzam Sleit, Manuel Alfonseca, Alfonso Ortega. Enhancing QuickSort Algorithm using a Dynamic Pivot Selection Technique // Wulfenia. 2012. Vol. 19, N 10. P. 543–552.