Двухуровневый параллельный алгоритм выполнения численной фазы разложения Холецкого для разреженных матриц*

С.А. Лебедев, И.Б. Мееров, Е.А. Козинов, Д.Р. Ахмеджанов, А.Ю. Пирова, А.В. Сысоев

Нижегородский государственный университет им. Н.И. Лобачевского

Рассматривается задача распараллеливания численной фазы разложения Холецкого для разреженных симметричных положительно определенных матриц. Предлагается новая схема распараллеливания мультифронтального метода для систем с общей памятью. Данная схема основана на сочетании двух подходов к организации параллелизма на разных уровнях дерева исключения. В нижней части дерева выполняется параллельная обработка узлов, хранящихся в приоритетной очереди. На верхних уровнях дерева узлы обсчитываются последовательно с использованием многопоточного BLAS. Результаты вычислительных экспериментов показывают сопоставимость выполненной реализации с решателями MUMPS и MKL PARDISO.

1. Введение

Системы линейных алгебраических уравнений (СЛАУ) с разреженной симметричной положительно определенной матрицей возникают при моделировании процессов во многих предметных областях. Огромная размерность таких систем (в современных приложениях – 10^7 и выше) приводит к большим затратам памяти и процессорного времени, что без сомнения позволяет отнести их решение к области применения высокопроизводительных вычислений. Прямые методы, основанные на факторизации матрицы системы, активно применяются для решения больших разреженных СЛАУ. В настоящее время в этой области разработан целый ряд параллельных алгоритмов для систем с различной архитектурой и соответствующие программные пакеты [1], среди которых широко распространены MKL PARDISO, MUMPS, SuperLU, CHOLMOD и другие.

С 2011 года на факультете ВМК ННГУ разрабатывается прямой решатель разреженных СЛАУ с симметричной положительно определенной матрицей, основанный на методе Холецкого [2]. В рамках данного решателя изучаются вопросы оптимизации соответствующих алгоритмов под современные многоядерные архитектуры. В данной работе рассматривается задача распараллеливания наиболее затратной по времени и памяти численной фазы разложения Холецкого. Несмотря на наличие множества алгоритмов и их реализаций, вопрос о развитии существующих методов построения масштабируемых алгоритмов и программных средств, ориентированных на системы с общей памятью, не потерял свою актуальность в связи с постоянным развитием многоядерных архитектур. Ранее в работе [3] мы предложили способ распараллеливания мультифронтального метода, основанный на динамической схеме балансировки нагрузки. В данной статье предлагается модификация, состоящая в сочетании возможностей базовой схемы с использованием параллельного BLAS для решения «тяжеловесных» задач на верхних уровнях дерева исключения. Будет показано, что применение данной комбинированной схемы позволяет улучшить масштабируемость программной реализации.

2. Постановка задачи и метод решения

Дана система линейных уравнений $Ax = b$, где A – разреженная симметричная положительно определенная матрица, b – плотный вектор, x– вектор неизвестных. Необходимо найти решение системы x.

* Работа выполнена при частичной поддержке гранта РФФИ №14-01-31455, гранта МОН РФ (соглашение от 27 августа 2013г. № 02.B.49.21.0003 между МОН РФ и ННГУ).

Принцип работы прямых методов основан на факторизации матрицы системы с последующим решением треугольных систем. Для симметричных положительно определенных матриц факторизация выполняется методом Холецкого. Для этого в большинстве случаев используется двухфазный подход: вначале находится портрет фактора, т.е. расположение ненулевых элементов (символьное разложение), затем полученный портрет заполняется значениями (численное разложение). Необходимо отметить, что символьная фаза выполняется гораздо быстрее численной, поэтому множество усилий исследователей направлено на оптимизацию и распараллеливание численной фазы.

Существует несколько методов выполнения численной фазы разложения Холецкого. Среди них можно выделить три наиболее широко используемых на практике: ориентированный влево (left-looking) [4], ориентированный вправо (right-looking) [4] и мультифронтальный (multifrontal) [5]. Основное отличие между ними заключается в способе формирования результирующей матрицы L, а также в способе хранения и размещении в памяти промежуточных результатов. Перечисленные методы показывают схожую производительность на различных тестовых наборах матриц, но с точки зрения многих исследователей мультифронтальный метод является наиболее перспективным для распараллеливания [6, 7]. По этой причине мультифронтальный метод используется в качестве базового в данной работе.

3. Краткое описание мультифронтального метода

Впервые мультифронтальный метод был представлен Даффом и Рейдом [8] в 1983 г., а затем развит Лю [9], Амстоем и Даффом [10]. К числу основных достоинств мультифронтального метода относят эффективное использование кэш-памяти всех уровней. При реализации с использованием подходящих структур данных основными становятся операции с плотными подматрицами, для выполнения которых может быть использован BLAS 3. Данный метод используется в одном из широко распространенных решателей с открытым исходным кодом MUMPS. К числу недостатков мультифронтального метода можно отнести высокие затраты памяти для представления промежуточных результатов и большое число операций с плавающей точкой, особенно для задач, полученных путем дискретизации трехмерного пространства [11]. Приведем краткое описание метода.

Численная фаза разложения Холецкого применяется для заполнения уже сформированного шаблона фактора матрицы численными значениями. Для этого в мультифронтальном методе процесс факторизации разбивается на факторизацию небольших плотных подматриц, называемых фронтальными (frontal matrix). При этом порядок получения столбцов фактора определяется графом задач, который в случае симметричной положительно определенной матрицы является деревом и называется деревом исключения (elimination tree). Каждый узел дерева соответствует столбцу матрицы. Таким образом, мультифронтальный метод может быть представлен как обход дерева исключения от листьев к корню. При посещении очередного узла происходит построение фронтальной матрицы, в результате частичной факторизации которой алгоритм формирует соответствующий столбец фактора. Для построения фронтальной матрицы используются значения соответствующего столбца исходной матрицы, а также матриц обновления, ассоциированных с детьми рассматриваемого узла в дереве исключения. После выполнения частичной факторизации фронтальной матрицы формируется столбец фактора и матрица обновления, которая будет использована при построении фронтальной матрицы родителя. Высокоуровневое описание мультифронтального метода приведено ниже (алгоритм 1). Символом $\oplus$ обозначена операция расширяющего сложения (extend add) [5].

Процедуры init_frontal_matrix, assembly_frontal_matrix и form_update_matrix реализуются с помощью вызовов соответствующих функций BLAS. Более подробное описание метода можно найти в работах [5, 11]. Последовательная реализация мультифронального метода в решателе ННГУ описана в работе [12].

Недостатком базовой версии численной факторизации является неэффективное использование кэш-памяти. Для решения этой проблемы на практике используются так называемые суперноды (supernode). Супернодом называется группа столбцов фактора, имеющих одинаковый шаблон ниже плотной треугольной подматрицы. Супернодальный мультифронтальный подход впервые был предложен Эшкрафтом, Гримсом, Льюисом, Пейтоном и Симоном [13] в 1987 г., а

затем исследован Нг и Пейтоном [14]. Суперноды позволяют формировать фактор по несколько столбцов за итерацию с использованием функций BLAS третьего уровня, что повышает эффективность использования кэш-памяти. Именно эта редакция мультифронтального метода используется в качестве базовой для данной работы.

Алгоритм 1. Высокоуровневое описание мультифронтального метода

```
1     foreach node i of elimination tree in topological order
2       init_frontal_matrix(F_i)
3           foreach son j of i do
4               U ← U ⊕ U_j
5           end for
6       assembly_frontal_matrix(F,U)
7       factorize(F)
8       form_update_matrix(U_i)
9       L_i←F_(1,*)
10    end for
```

4. Схемы распараллеливания

В мультифронтальном методе могут быть использованы два способа организации параллелизма: применение параллельных функций BLAS и параллельное решение независимых друг от друга задач в соответствии со структурой дерева исключения. Рассмотрим перспективы применения этих методов.

4.1 Использование параллельного BLAS

Большая часть вычислений в мультифронтальном методе приходится на процедуры BLAS, такие как умножение плотных матриц и решение плотных систем линейных уравнений с треугольной матрицей. Поэтому использование существующих библиотек для высокопроизводительных вычислений, таких как, например, Intel MKL, является самым естественным способом распараллеливания численной фазы разложения Холецкого. К сожалению, эксперименты показывают, что применение этого подхода чаще всего приводит к разочаровывающим результатам. Этот факт объясняется тем, что большинство вспомогательных матриц, возникающих в ходе выполнения алгоритма, имеют маленькую размерность и поэтому накладные расходы, связанные с организацией параллелизма не компенсируются последующей параллельной обработкой. Таким образом, необходимую производительность можно получить, только если в качестве основного метода распараллеливания использовать распараллеливание по дереву исключения.

4.2 Распараллеливание по дереву исключения

Распараллеливание мультифронтального метода может быть выполнено на основе дерева исключения T, содержащего информацию о зависимостях по данным, возникающих в ходе вычислений. Пусть $T[k]$ – часть дерева исключения с корнем в вершине k. Показано [11], что два столбца i и j фактора L могут быть вычислены параллельно тогда и только тогда, когда поддеревья $T[i]$ и $T[j]$ не пересекаются, то есть не имеют общих узлов. Этот результат является основной для построения алгоритмов параллельной факторизации. При этом в качестве единицы работы (*задачи*) можно рассматривать построение фронтальной матрицы, соответствующей очередному узлу дерева. Основная проблема, возникающая при разработке параллельной редакции метода, заключается в наличии существенного дисбаланса между вычислительной трудоемкостью возникающих *задач*. Решение данных задач предполагает выполнение операций над матрицами, размеры которых могут отличаться на порядки от узла к узлу при сохранении общей тенденции укрупнения матриц при перемещении от листьев к корню. Для решения про-

блемы балансировки нагрузки могут использоваться как статические, так и динамические схемы.

4.3 Статические схемы распараллеливания

Существует множество методов, использующих статическую схему распараллеливания, однако большинство из них были разработаны для систем с распределенной памятью. В последнее время предпринимаются усилия [15] для переноса одного из наиболее эффективных методов статического распараллеливания – алгоритма Гейста-Нг [16] для работы в системах с общей памятью. Идея алгоритма заключается в нахождении некоторого слоя в дереве исключения, то есть множества узлов, которые не обязательно находятся на одном уровне, но при этом не имеют общих потомков. Найденный слой должен обладать свойством сбалансированности, так чтобы количество операций, необходимых для обработки узлов поддеревьев с корнями в узлах найденного слоя, удовлетворяло заданному порогу и было примерно одинаковым. При реализации алгоритма свойство сбалансированности можно понимать как число операций с плавающей точкой, либо как оценку времени, необходимого для обработки узла.

На рисунке 1 приведен пример работы алгоритма. Найденный слой представляет срез дерева в узлах 6, 11, 14. Выделенные цветом поддеревья могут быть обработаны параллельно.

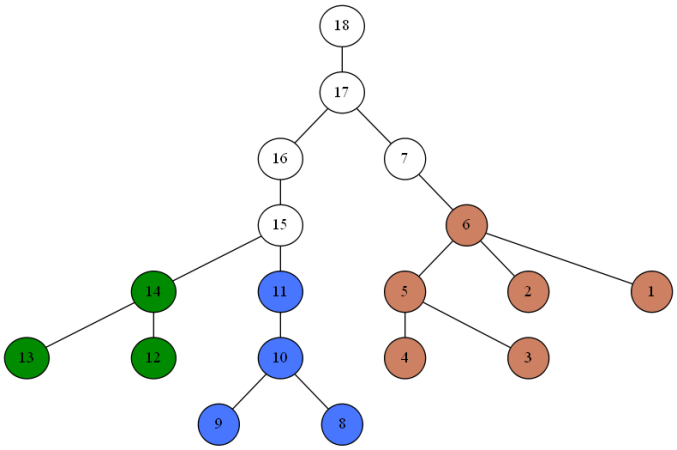

Рис. 1. Пример разбиения дерева исключения с использованием алгоритма Гейста-НГ. Разными цветами отмечены поддеревья, которые могут быть обработаны параллельно

5. Двухуровневый параллельный алгоритм

Одним из основных недостатков статических схем является невозможность достаточно точно оценить объем работы, необходимый для обработки каждого узла дерева исключения. Поэтому в данной работе предлагается другой способ балансировки нагрузки, основанный на динамической схеме.

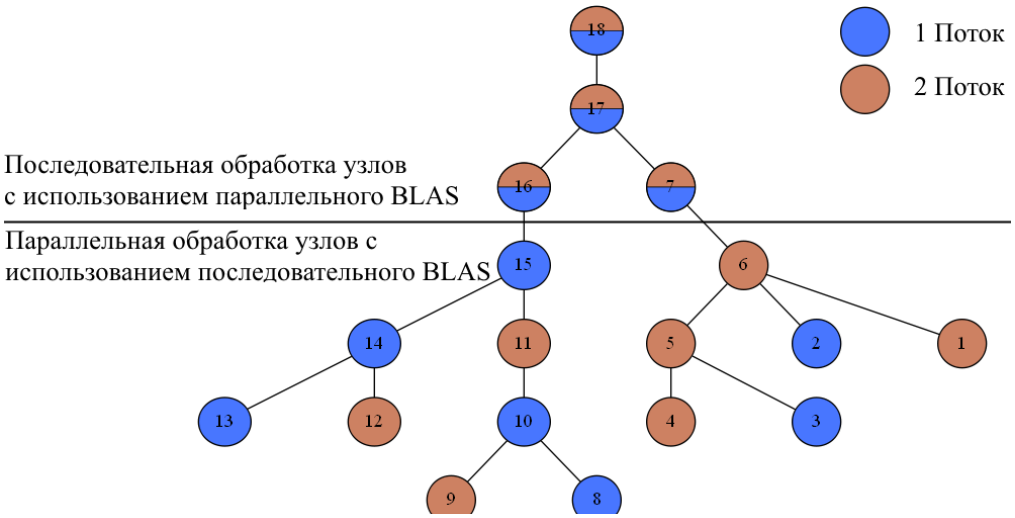

Рис. 2. Пример работы мультифронтального метода на основе двухуровневой схемы распараллеливания

В рамках динамической схемы строится пул задач и на каждом шаге алгоритма поток достает задачу из очереди и приступает к ее выполнению. Каждая задача соответствует вычислению соответствующего столбца фактора и состоит из четырех подзадач: вычисление матрицы узла, вычисление фронтальной матрицы, формирование из фронтальной матрицы столбца фактора, вычисление матрицы обновления. Пул задач организуется в виде приоритетной очереди.

Для формирования очереди задач используется алгоритм, который учитывает различные характеристики узлов дерева для достижения лучшей балансировки [3]. Алгоритм обходит все узлы дерева в соответствии с топологической перестановкой, сформированной раннее, и на каждой итерации цикла добавляет рассматриваемый узел в очередь. Приоритет узла в очереди складывается из основного и второстепенного, где первый отвечает за правильный обход столбцов в параллельном мультифронтальном методе, а второй – за улучшение балансировки.

Основной приоритет равен количеству детей узла в дереве исключения, а второстепенный вычисляется как оценка трудоемкости решения соответствующих подзадач. Трудоемкость решения задачи оценивается как количество операций с плавающей точкой.

Представленная схема позволяет эффективно использовать вычислительные ресурсы на нижних уровнях дерева исключения, но при приближении к корню возможности для параллелизма по задачам становятся ограниченными, при этом размеры обрабатываемых матриц значительно увеличиваются. В этот момент целесообразно изменить схему распараллеливания таким образом, чтобы вычислительные потоки использовались внутри вызовов многопоточные реализации функций BLAS (алгоритм 2).

Алгоритм 2. Параллельный мультифронтальный метод на основе двухуровневой схемы

```
1     procedure process_node(node i of elimination_tree)
2       init_frontal_matrix(F_i)
3       foreach son j of i do
4         U ← U ⊕ U_j
5       end for
6       assembly_frontal_matrix(F,U)
7       factorize(F)
8       form_update_matrix(U_i)
9       L_i←F_(1,*)
10    end procedure
11
12    procedure two-level_parallel_multifrontal
13      omp_set_num_threads(MAX_SYSTEM_THREADS);
14      blas_set_num_threads(1);
15
16      #pragma omp parallel
```

```
17          while(task_queue.hasTaskET())
18                  #pragma omp critical (queue)
19                      i←task_queue.get_task_with_highest_priority();
20                  process_node(i)
21                  #pragma omp critical (queue)
22                      task_queue.increase_task_primary_priority(parent(i))
23          end while
24
25        while(task_queue.hasTask())
26          i←task_queue.get_task_with_highest_priority();
27          process_node(i)
28          task_queue.increase_task_primary_priority(parent(i))
29        end while
30      end procedure
```

6. Результаты вычислительных экспериментов

6.1 Тестовая инфраструктура

Результаты экспериментов получены с использованием узла кластера, содержащего два восьмиядерных процессора Intel Sandy Bridge E5-2660 2.2 GHz, 64GB RAM, работающего под управлением ОС Linux CentOS 6.4. Использовался компилятор Intel C++ Compiler и библиотека Intel MKL BLAS из пакета Intel® Parallel Studio XE 2013 SP1. Для проведения экспериментов были выбраны матрицы из коллекции университета Флориды [17]. Характеристики тестовых матриц представлены ниже (таблица 1). Все они являются симметричными положительно определенными. В качестве перестановок, уменьшающих заполнение фактора, использовался METIS [18], но также могут быть использованы другие переупорядочиватели [19, 20].

Таблица 1. Характеристики тестовых матриц

Название матрицы	Порядок	Число ненулевых элементов	Число ненулевых элементов в факторе
Pwtk	217 918	5 926 171	49 025 872
Msdoor	415 863	10 328 399	51 882 257
parabolic_fem	525 825	2 100 225	25 571 376
tmt_sym	726 713	2 903 835	28 657 615
boneS10	914 898	28 191 660	266 173 272
Emilia_923	923 136	20 964 171	1 633 654 176
audkiw_1	943 695	39 297 171	1 225 571 121
bone010_M	986 703	12 437 739	363 650 592
bone010	986 703	36 326 514	1 076 191 560
ecology2	999 999	2 997 995	35 606 934
thermal2	1 228 045	4 904 179	50 293 930
StocF-1465	1 465 137	11 235 263	1 039 392 123
Hook_1498	1 498 023	31 207 734	1 507 528 290
Flan_1565	1 564 794	59 485 419	1 451 334 747
G3_circuit	1 585 478	4 623 152	90 397 858

6.2 Использование параллельных библиотек BLAS

Наиболее простым способом распараллеливания мультифронтального метода является использование высокопроизводительных параллельных библиотек BLAS. Для этого не требуется изменять исходный код, необходимо лишь собрать его с соответствующей реализацией BLAS. На диаграмме (рисунок 3; слева) показано ускорение численной фазы разложения Холецкого при запуске с различным числом потоков BLAS.

Каждый многоугольник на рисунке соответствует запуску с определенным количеством потоков. По осям отложено время работы на соответствующей матрице. Для матрицы Hook_1498 указано время работы в 8 потоков (при работе в 16 потоков запуск завершается с ошибкой из-за нехватки памяти).

Исходя из результатов запусков версии с параллельной библиотекой BLAS (рисунок 3; слева) можно сделать следующие выводы. Все тестовые матрицы можно разделить на 2 группы в зависимости от степени их заполненности. Для первой группы, где матрицы больше и плотность их выше (матрицы Flan_1565, Emilia923, audikw1, bone010, StocF-1465, Hook_1498), использование параллельной библиотеки BLAS позволяет получить ускорение до 6 раз при запуске в 16 потоков. Для второй группы, где матрицы либо маленькие, либо сильно разреженные (матрицы G3_circuit, pwtk, msdoor, parabolic_fem, tmt_sym, boneS10, bone010_M, ecology2, thermal2) ускорения при увеличении числа потоков BLAS практически не наблюдается. Также стоит отметить, что MKL BLAS имеет встроенный контроль размера матриц и не производит параллельную обработку, если размер матрицы слишком маленький. Таким образом, отсутствие ускорения на матрицах из второй группы можно считать хорошим результатом, так как при отсутствии подобного контроля за размером матрицы можно было бы наблюдать замедление.

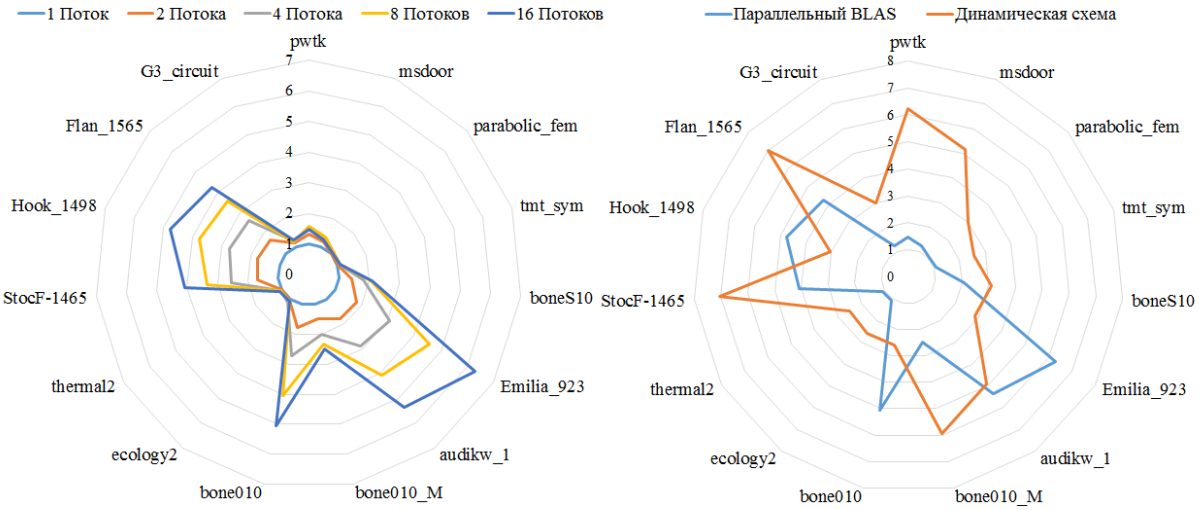

Рис. 3. Сравнение ускорения численной фазы разложения Холецкого:
при запуске с разным числом потоков BLAS (слева); при запуске в 16 потоков с использованием параллельного BLAS и динамической схемы (справа)

В работе [3] мы предложили использовать динамическую схему распараллеливания мультифронтального метода. Сравнивая ускорение с использованием различных схем распараллеливания (рисунок 3; справа) можно видеть, что для 4 из 6 матриц первой группы (Emilia923, audikw1, bone010, Hook_1498) использование параллельного BLAS дает лучшее ускорение в среднем в 1.7 раза. Тем не менее, для остальных матриц предпочтительнее использовать динамическую схему. Этот факт говорит о том, что комбинация указанных методов распараллеливания (алгоритм 2) может дать потенциально лучшее ускорение по сравнению с каждой схемой в отдельности, что в дальнейшем подтверждается вычислительными экспериментами.

6.3 Выбор момента переключения между схемами распараллеливания

Важнейшим элементом алгоритма, влияющим на время работы численной фазы при использовании двухуровневой схемы распараллеливания, является критерий переключения между схемами (по узлам дерева; в рамках одного узла при помощи BLAS). Анализ параллельных запусков динамической схемы распараллеливания, использующей параллелизм на уровне логических задач в сочетании с *последовательным BLAS*, показал, что основным фактором, ограничивающим итоговое ускорение, является недостаток свободных задач, начиная с некоторого момента подъема по дереву исключения. В связи с этим предлагается использовать в качестве критерия сравнение числа необработанных задач с некоторым пороговым значением, выбирае-

мым экспериментально. После срабатывания данного критерия происходит переход к последовательной обработке узлов дерева с использованием параллельного BLAS (алгоритм 2).

Для изучения вопроса о выборе порогового значения были выбраны 4 матрицы, представляющие 4 возможных класса: «небольшие разраженные» (parabolic_fem), «небольшие плотные» (pwtk), «большие разреженные» (G3_Circuit) и «большие плотные» (audikw_1).

Результаты приведены на диаграмме (рисунок 4). По горизонтальной оси отложено пороговое значение, а по вертикальной – время работы двухуровневого алгоритма, отнесенное к времени работы с использованием динамической схемы (в зависимости от матрицы). Во всех запусках использовалось 16 вычислительных ядер. Значения, меньшие единицы, соответствуют преимуществу двухуровневой схемы над обычной.

Для небольших более плотных матриц и больших сильно разреженных время работы при изменении порога не изменяется, кроме того оно практически совпадает с временем работы при распараллеливании с использованием динамической схемы, отношение времен колеблется около единицы. Для больших более плотных матриц наблюдается значительное ускорение относительно динамической схемы, причем оно наблюдается уже при небольших значениях параметра и в дальнейшем практически не меняется.

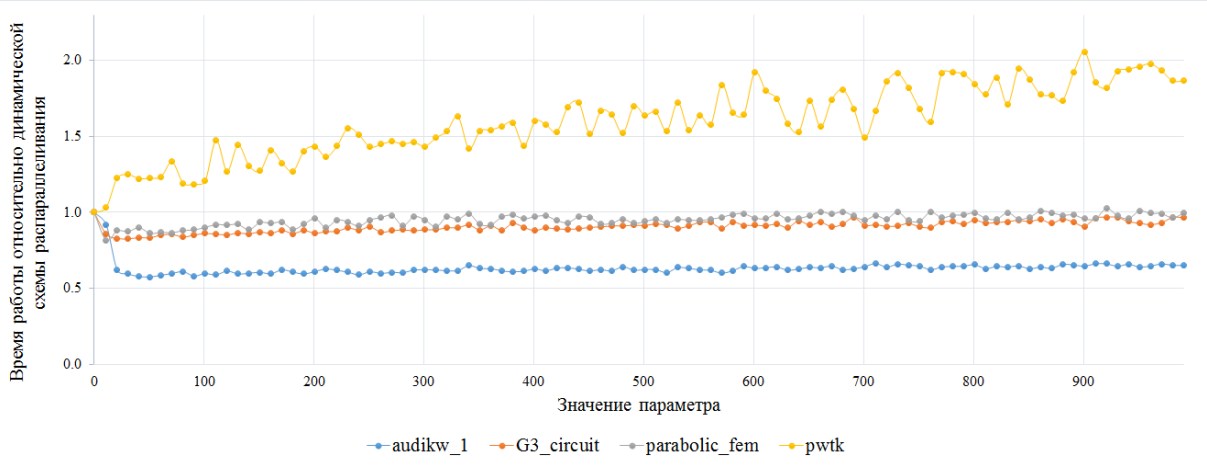

Рис. 4. Зависимость времени работы двухуровневого алгоритма
от момента переключения между двумя параллельными схемами

Этот факт говорит о том, что большая часть работы сосредоточена в небольшой окрестности корня дерева, где параллелизм по задачам ограничен, но есть ресурс для использования параллельных библиотек BLAS. Ниже этой окрестности фронтальных матриц больше, они имеют средний размер, и использование параллелизма по задачам и параллелизма BLAS дает схожий результат. Для матриц из последней группы ситуация выглядит противоположным образом. Размер фронтальных матриц настолько мал, что использование параллельного BLAS сразу же приводит к замедлению. Однако таким замедлением можно пожертвовать, так как абсолютное время работы составляет менее 1 секунды.

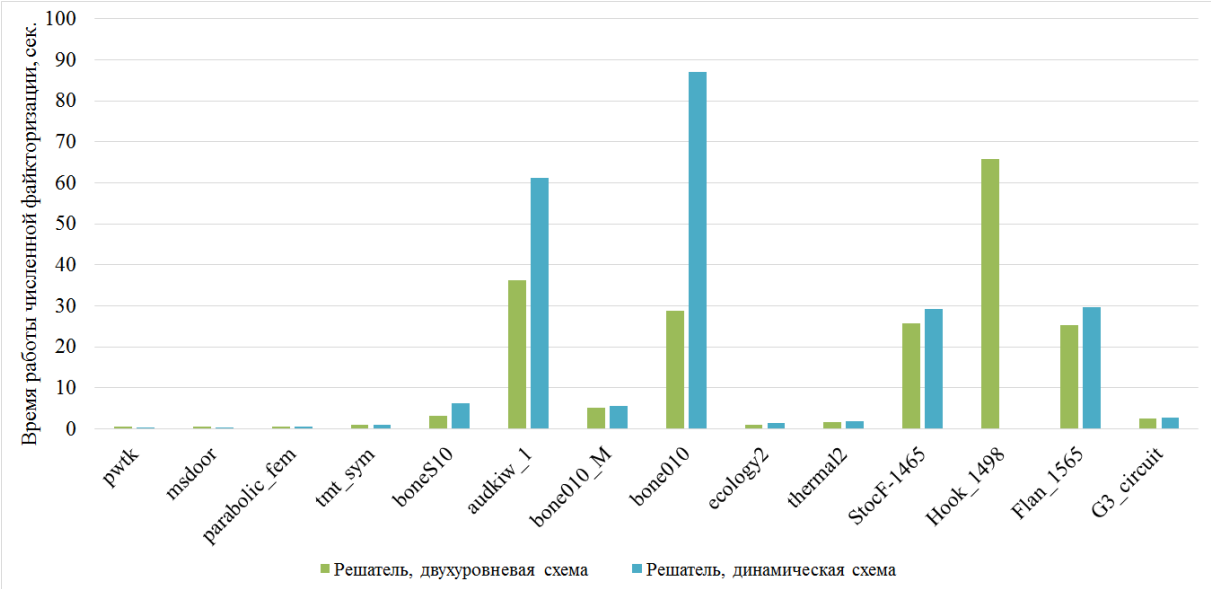

Рис. 5. Сравнение времени работы мультифронтального метода при использовании динамической и двухуровневой схем

На диаграмме (рисунок 5) приведено абсолютное время работы мультифронтального метода с использованием различных схем распараллеливания. Во всех запусках использовались 16 ядер. Значение порога переключения схем для всех матриц было выбранным одинаковым и равнялось 200. Можно видеть, что для всех представленных матриц, время работы на которых превышает 5 секунд, новый двухуровневый метод показывает лучшие результаты. Данный эффект проявляется наиболее явно на матрице bone010, где удалось получить ускорение в 3 раза.

Отдельного внимания заслуживает вопрос об объеме используемой памяти. Так, фронтальные матрицы на верхних уровнях дерева исключения имеют больший размер. В связи с этим, при использовании параллелизма на задачах для обработки узлов верхних уровней требуется хранение вспомогательных структур данных для каждого из 16 потоков, что приводит к большим затратам памяти. Напротив, в двухуровневой схеме фронтальные матрицы узлов верхних уровней обрабатываются потоками совместно, что позволяет значительно сократить размер вспомогательных структур данных. В частности, применение описанных методов позволило получить правильное решение для матрицы Hook_1498 при использовании 16 потоков, в отличие от базовой динамической схемы.

6.4 Сравнение с известными решателями

Был проведен ряд экспериментов на тестовых матрицах с использованием следующих известных и широко распространенных решателей:

- MKL PARDISO из Intel Math Kernel Library в составе Intel Parallel Studio XE 2013 SP1 [21];
- MUMPS (лучшее время работы из двух актуальных версий ver. 4.10.0, ver 5.0.0) [22].

Для всех пакетов использовались одинаковые перестановки, полученные с помощью METIS, а также функции BLAS и ScaLAPACK из библиотеки Intel MKL. Полученные результаты приведены на диаграмме (рисунок 6).

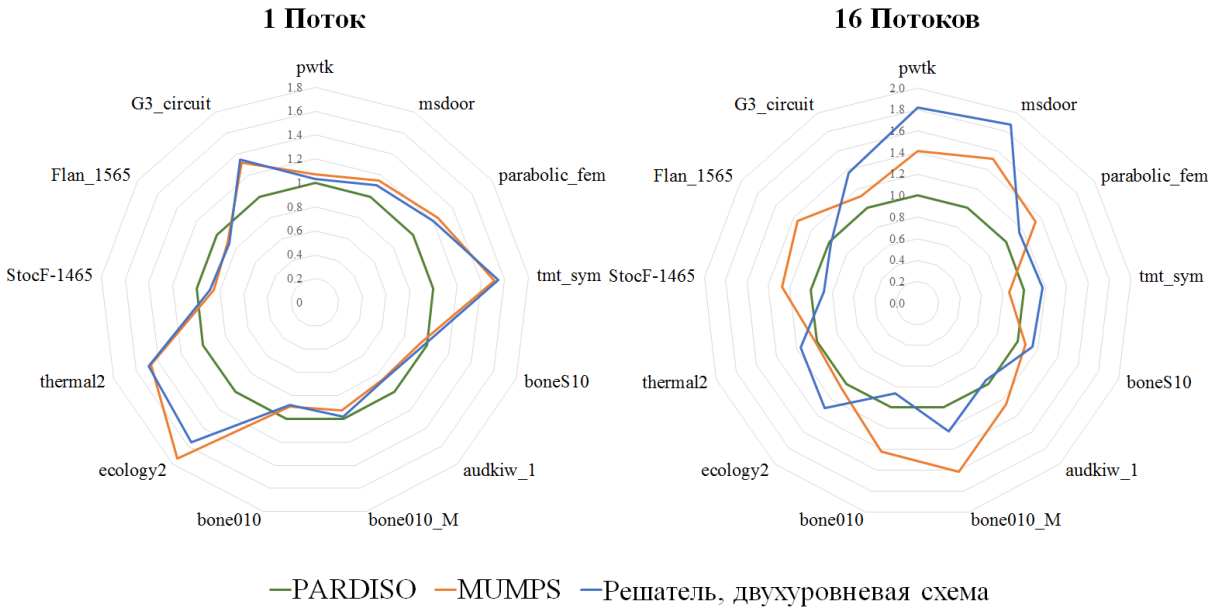

Рис. 6. Сравнение численных фаз решателей. По осям отложено время работы. За единицу взято время работы MKL PARDISO

Из результатов можно сделать следующие выводы. Для запусков в 1 поток реализованный мультифронтальный метод показывает схожие результаты с MUMPS. Это объясняется тем фактом, что в обоих решателях в качестве метода численного разложения используется мультифронтальный метод. В тоже время оба решателя проигрывают PARDISO на 6 матрицах (msdoor, tmt_sym, ecology2, thernaml2, G3_circuit, parabolic_fem), на 4 матрицах (audikw_1, bone010, StocF-1465, Flan_1565) заметен выигрыш, в остальных случаях время работы сопоставимо. Наибольший выигрыш разработанного решателя по сравнению с PARDISO получен на матрице audikw_1 и составляет 14%, а наибольшее отставание, в 1.5 раза, на матрице ecology2.

Рассматривая результаты экспериментов для запусков решателей в 16 потоков можно видеть, что PARDISO сохраняет преимущество во времени работы и показывает лучшие результаты на 5 матрицах. Сравнивая двухуровневый алгоритм и MUMPS, нужно отметить преимущество первого на 6 матрицах (parabolic_fem, audikw_1, bone010_M, bone010, StocF-1465, Flan-1565). В свою очередь MUMPS также выигрывает на 6 матрицах (pwtk, msdoor, tmt_sym, ecology2, thermal2, G3_circuit). Однако, если обратиться к диаграмме (рисунок 5), можно видеть, что время работы на этих матрицах достаточно маленькое по сравнению с матрицами, на которых получен выигрыш. Сравнивая время работы двухуровневого алгоритма и PARDISO можно отметить, что ситуация во многом схожая: матрицы, на которых отставание наиболее заметно (pwtk, msdoor, ecology2, thermal2, G3_circuit), обрабатываются численно фазой достаточно быстро и по причинам, описанным в предыдущем разделе, дают худшее масштабирование. В остальных запусках время работы численной фазы обоих решателей в большей степени сопоставимо.

7. Заключение

Основным результатом работы является комбинированная схема распараллеливания мультифронтального метода. Данная схема сочетает лучшие свойства двух подходов к организации параллелизма при обработке дерева исключения. Так, большое число «легковесных задач», соответствующих нижним уровням дерева, решается в рамках парадигмы распараллеливания по задачам с динамической балансировкой нагрузки. При этом на верхних узлах дерева малое число «тяжеловесных задач» решается путем применения многопоточного BLAS. В работе сформулирован критерий переключения между схемами, показан выигрыш в производительности и памяти по сравнению с ранее подготовленной реализацией, выполнено сравнение с MKL PARDISO и MUMPS.

В дальнейшем планируется рассмотреть возможность усовершенствования разработанных программных реализаций за счет сокращения накладных расходов на организацию параллелизма. В частности, представляют интерес перспективы применения разных реализаций приоритетных очередей. Другим направлением дальнейших исследованием является разработка гетерогенных реализаций, использующих для расчетов не только традиционные процессоры, но и ускорители вычислений.

Литература

1. Li X. Direct Solvers for Sparse Matrices. URL: http://crd-legacy.lbl.gov/~xiaoye/SuperLU/SparseDirectSurvey.pdf (дата обращения: 15.06.2015).
2. Козинов Е.А. Лебедев И.Г. Лебедев С.А. Малова А.Ю. Мееров И.Б. Сысоев А.В. Филиппенко С.С. Новый решатель для алгебраических систем разреженных линейных уравнений с симметричной положительно определенной матрицей // Вестник Нижегородского университета им. Н.И. Лобачевского. – Н. Новгород: Изд-во ННГУ, 2012. – №5(2) – С. 376-384.
3. Lebedev S., Akhmedzhanov D., Kozinov E., Meyerov I, Pirova A., Sysoyev A. Dynamic Parallelization Strategies for Multifrontal Sparse Cholesky Factorization //Parallel Computing Techologies. – Springer LNCS, 2015 (принята к печати).
4. Davis Timothy A. Direct methods for sparse linear systems. Vol. 2. Siam, 2006.
5. Liu J. W. H. The multifrontal method for sparse matrix solution: Theory and practice // SIAM review. 1992. Vol. 34, No. 1. P. 82-109.
6. Kalinkin A., Arturov K. Asynchronous approach to memory management in sparse multifrontal methods on multiprocessors // Applied Mathematics. 2013. Vol. 4, No. 12A. P. 33-39.
7. George T., Saxena V., Gupta A., Singh A., Choudhury A. R. Multifrontal factorization of sparse SPD matrices on GPUs // Parallel & Distributed Processing Symposium (IPDPS), 2011. P. 372-383.
8. Duff I.S., Reid J.K. The multifrontal solution of indefinite sparse symmetric linear // ACM Transactions on Mathematical Software (TOMS). 1983. Vol. 9, No. 3. P. 302-325.
9. Liu J.W. The multifrontal method and paging in sparse Cholesky factorization // ACM Transactions on Mathematical Software (TOMS). 1989. Vol. 15, No. 4. P. 310-325.
10. Amestoy P.R., et al. Vectorization of a multiprocessor multifrontal code // International Journal of High Performance Computing Applications. 1989. Vol 3, No. 3. P. 41-59.
11. L'Excellent J.Y.: Multifrontal Methods: Parallelism, Memory Usage and Numerical Aspects // Ph.D. thesis, Ecole normale superieure de lyon-ENS LYON. 2012.
12. Лебедев С. А., Козинов Е. А. Разработка нового решателя разреженных систем линейных уравнений // Высокопроизводительные параллельные вычисления на кластерных системах: Материалы XIII Всероссийской конференции. / Изд-во ННГУ: Н. Новгород, 2013. С. 301-306.
13. Ashcraft C.C., Grimes R.G., Lewis J.G., Peyton B.W., Simon H.D., Bjorstad P.E. Progress in sparse matrix methods for large linear systems on vector supercomputers // International Journal of High Performance Computing Applications. 1987. Vol 1, No. 4. P. 10-30.
14. Ng E., Peyton B.W. A supernodal Cholesky factorization algorithm for shared-memory multiprocessors // SIAM Journal on Scientific Computing. 1993. Vol 14, No. 4. P. 761-769.
15. L'Excellent J.Y., Sid-Lakhdar M.W. Introduction of shared-memory parallelism in a distributed-memory multifrontal solver // Research Report. 2013. RR-8227 <hal-00786055>.
16. Geist G., Ng E. Task scheduling for parallel sparse Cholesky factorization // International Journal of Parallel Programming. 1989. Vol. 18, No. 4. P. 291-314.

17. Davis T.A., Hu Y. The university of Florida sparse matrix collection // ACM Transactions on Mathematical Software (TOMS). 2011. Vol 38, No. 1.
18. Karypis G., et al. A Fast and Highly Quality Multilevel Scheme for Partitioning Irregular Graphs // SIAM Journal on Scientific Computing. 1999. Vol. 20, No. 1. P. 359-392.
19. Pellegrini F. Scotch and libScotch 6.0 User's Guide // Tech. rep., LaBRI. 2012.
20. Pirova A., Meyerov I. MORSy – a new tool for sparse matrix reordering // In Proceedings of an International Conference on Engineering and Applied Sciences Optimization. 2014. P. 1952-1963.
21. Intel Math Kernel Library Reference Manual.
URL: [https://software.intel.com/en-us/node/470282] (дата обращения 15.06.2015).
22. MUltifrontal Massively Parallel Solver (MUMPS 5.0.0) User's guide
URL: [http://mumps.enseeiht.fr/doc/userguide_5.0.0.pdf] (дата обращения 15.06.2015).