Разработка прямого решателя разреженных СЛАУ для систем с распределенной памятью

С.А. Лебедев

Нижегородский государственный университет им. Н. И. Лобачевского

Решение разреженных систем линейных алгебраических уравнений (СЛАУ) лежит в основе многих задач вычислительной физики, химии, биологии. Часто на практике приходится выбирать между прямыми и итерационными методами, при этом каждый из подходов имеет свои преимущества и недостатки. Схема прямого решателя для разреженных СЛАУ основывается на факторизации исходной матрицы и состоит из следующих основных этапов: переупорядочивание, символьное разложение (фаза анализа), численное разложение, обратный ход метода Гаусса. Среди перечисленных этапов наиболее затратным с точки зрения времени работы является этап численной факторизации, задачей которого является вычисление значений ненулевых элементов фактора.

Одним из основных недостатков прямых методов решения разреженных СЛАУ является большой объем потребляемой памяти на всех этапах работы алгоритма. Поэтому создание программного обеспечения, способного потенциально задействовать множество вычислительных узлов, является важной и актуальной задачей. Одним из стандартов программирования для распределенной памяти является MPI. Библиотека MPI позволяет создавать независимо работающие процессы, назначать работу каждому из них, а также обмениваться данными посредством сообщений. Такой подход не только обеспечивает возможность разным вычислительным процессам работать в рамках распределенных систем, но и потенциально может улучшить работу программы в рамках одного узла за счет улучшения локальности данных. Этот эффект достигается за счет использования гибридной модели программирования MPI и OpenMP. Именно в этом направлении выполняется данная работа.

Один из первых способов распараллеливания прямых методов для систем с распределенной памятью был предложен в 1989 году Джорджем, Лю и др. [1]. Среди некоммерческих распределенных прямых решателей широкое распространение в настоящее время получил программный пакет MUMPS [2]. Изначально библиотека поддерживала работу только для систем с распределенной памятью, для балансировки вычислений между узлами использовала схемы на основе алгоритма Гейста-Нг [3]. Такие схемы работают следующим образом. Рассмотрим слой дерева исключения. Каждый узел этого слоя, а также поддерево с корнем в текущем узле обрабатывается отдельным процессом. Затем все остальные узлы, начиная с рассматриваемого слоя и до корня дерева, могут быть обработаны параллельно таким образом, что фронтальные матрицы делятся на полосы по строкам, а каждая полоса обрабатывается отдельным процессом. В конце вычислений, если корень дерева достаточно большой, соответствующая ему группа столбцов обрабатывается с использованием распределенных реализаций библиотек BLAS и ScaLapack [4]. В настоящее время функциональность решателя MUMPS существенно расширена. Так, в библиотеку добавлена поддержка параллелизма для систем с общей памятью, использующая различные способы распараллеливания, в частности, схемы, основанные на предсказании времени работы или числа операций с плавающей запятой [5].

Данная работа продолжает серию статей [6,7 и др.], посвященную разработке масштабируемых алгоритмов распараллеливания вычислений в процессе прямого решения разреженных симметричных СЛАУ. Ранее при разработке соответствующих алгоритмов для систем с общей памятью удалось добиться результатов, сопоставимых по производительности с решателями MUMPS и PARDISO. В данной работе предлагается новый параллельный алгоритм, ориентированный на кластерные системы и сочетающий параллелизм на уровне процессов (на базе MPI) и параллелизм на уровне потоков (на базе OpenMP или TBB), и использующий предложенные ранее алгоритмические улучшения для систем с общей памятью. Для распараллеливания численного этапа прямого решателя используется следующий подход (рисунок 1). Расчеты распараллеливаются по дереву исключения, которое описывает зависимости между вычислениями. Перед началом работы метода дерево исключения делится на независимые поддеревья, которые можно

обработать параллельно. Количество поддеревьев должно быть не меньше, чем количество MPI-процессов. Если количество поддеревьев больше этого числа, то в этом случае некоторым MPI-процессам будет назначено более одного поддерева. Разбиение на поддеревья выполняется с учетом объема вычислений, требуемых для обработки каждого узла дерева исключения, при этом метод разбиения пытается достичь приемлемой балансировки за счет использования жадного алгоритма назначения задач. Такая схема работы основана на часто применяемом подходе Гейста-Нг. Как было указано ранее, для обработки узлов дерева исключения, находящихся выше найденного слоя, используются параллельные алгоритмы для общей памяти, различные варианты реализации которого представлены в работах [6, 7].

Описание алгоритма
Вход: матрица A, портрет матрицы L
Выход: матрица L
Процедура: параллельный алгоритм численного разложения
ШАГ 1. Найти уровень в дереве исключения, такой что отношение
max(TCE для всех поддеревьев)/min(TCE для всех поддеревьев) было минимальным. Здесь TCE – оценка количества операций с плавающей запятой, требуемых для обработки поддерева.
ШАГ 2. Назначить поддеревья вычислительным процессам так, чтобы объем работы у каждого процесса был по возможности равным.
ШАГ 3. Каждый процесс обрабатывает свои поддеревья в соответствии с параллельным алгоритмом для систем с общей памятью.
ШАГ 4. Мастер-процесс собирает информацию с остальных вычислительных процессов и выполняет обработку оставшихся узлов в соответствии с параллельным алгоритмом для систем с общей памятью.

Рисунок 1. Высокоуровневое описание параллельной реализации численного разложения прямого решателя разреженных СЛАУ с симметричной положительно определенной матрицей для кластерных систем

В настоящее время выполнена базовая реализация указанного подхода, проверена ее корректность, получены первые результаты производительности. В рамках постерного доклада будут представлены результаты экспериментов на матрицах коллекции SuiteSparse (ранее – коллекция разреженных матриц университета Флориды). Будет выполнено сравнение производительности с MUMPS и MKL PARDISO, анализ масштабируемости на больших матрицах, проведены эксперименты по выбору оптимального режима запуска (число MPI-процессов и OpenMP-потоков).

Литература

1. George A. et al. Solution of sparse positive definite systems on a hypercube //Journal of Computational and Applied Mathematics. – 1989. – Т. 27. – №. 1-2. – С. 129-156.
2. MUltifrontal Massively Parallel Solver (MUMPS 4.10.0) User's guide // Technical report ENSEEIHT-IRIT. – 2011. URL: [http://mumps.enseeiht.fr/doc/userguide_4.10.0.pdf]
3. Geist G. A., Ng E. Task scheduling for parallel sparse Cholesky factorization //International Journal of Parallel Programming. – 1989. – Т. 18. – №. 4. – С. 291-314.
4. L'Excellent J. Y. Multifrontal methods: parallelism, memory usage and numerical aspects : дис. – Ecole normale supérieure de Lyon, 2012.
5. Sid Lakhdar M. W. Scaling the solution of large sparse linear systems using multifrontal methods on hybrid shared-distributed memory architectures : дис. – Lyon, École normale supérieure, 2014.
6. Лебедев С. А., Мееров И. Б., Козинов Е. А., Ахмеджанов Д. Р., Пирова А. Ю., Сысоев А. В. Двухуровневый параллельный алгоритм выполнения численной фазы разложения Холецкого для разреженных матриц // Суперкомпьютерные дни в России: Труды международной конференции (28–29 сентября 2015 г., г. Москва). — М.: Изд-во МГУ, 2015. — С. 133–144.

7. Lebedev S. et al. Dynamic parallelization strategies for multifrontal sparse Cholesky factorization //Int. Conference on Parallel Computing Technologies. – Springer, Cham, 2015. – С. 68-79.