УДК 004.4’422

АВТОМАТИЗАЦИЯ РАСПАРАЛЛЕЛИВАНИЯ И ПЕРЕНОСИМОСТИ ПРОГРАММ ДЛЯ СОВРЕМЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Б.Я. Штейнберг
Южный федеральный университет
Россия, 344004, Ростов-на-Дону, Б. Садовая ул., 105
E-mail: borsteinb@mail.ru

Ключевые слова: автоматизация распараллеливания, оптимизирующий компилятор, распределенная память, суперкомпьютер на кристалле, переносимость программ с сохранением эффективности.

Аннотация: В представленной работе описан проект системы преобразования программ для автоматизации распараллеливания на различные вычислительные системы, в том числе, с распределенной памятью. Система предназначена для создания распараллеливающих компиляторов на вычислительные системы с распределенной памятью. В первую очередь, проект ориентирован на микросхемы типа «суперкомпьютер-на-кристалле». Предполагается связь данного проекта с компилирующей системой LLVM. Обсуждаются вопросы минимизации межпроцессорных пересылок при распараллеливании. Основной подход к созданию таких компиляторов – блочно-аффинные размещения данных в распределенной памяти. Показано, что предлагаемая система преобразования программ должны создаваться на основе высокоуровневого внутреннего представления.

1. Введение

В данной работе обсуждается проект системы преобразования программ (СПП) для повышения эффективности использования современных вычислительных систем.

Производительность программно-аппаратных комплексов существенна для многих задач экономики. Например, компьютерные системы управления беспилотным транспортом должны быстро обрабатывать поступающий к ним поток информации. От этой быстроты зависит и скорость транспорта, и возможность анализа большого множества встречаемых на пути объектов, и точность принятия решений.

Для ускорения существующей высокопроизводительной программы иногда возникает необходимость ее перенести на новый более быстрый компьютер. При этом на новом компьютере программа должна эффективно использовать возможности этого компьютера. Такую эффективность естественно измерять долей пиковой производительности компьютера.

Часто можно встретить мнение о том, что производительность программы зависит от степени ее распараллеливания. Однако, это не всегда так, иногда распараллеливание замедляет программу. По этой причине написанные на OpenCL параллельные программы иногда при переносе с одной микросхемы на другую могут терять эффективность (долю пиковой производительности) в несколько раз [1]. Это

объясняется тем, что разные компьютеры могут иметь разные структуры памяти (включая иерархию кэш-памяти), а время обращения к памяти на современных компьютерах существенно превышает время вычислительных операций.

Решением проблемы эффективной переносимости с одного компьютера на другой может быть наличие эффективных распараллеливающих компиляторов на каждом из двух компьютеров: исходный высокоуровневый текст программы следует перекомпилировать на втором компьютере.

В последнее время появились такие многоядерные процессоры, у которых в локальной памяти каждого вычислительного ядра могут задерживаться данные без их сбрасывания в оперативную память. При этом вычислительные ядра могут обмениваться данными по расположенной на микросхеме коммуникационной сети (network-on-chip). Такая архитектура избавляет от длительной операции обращения к оперативной памяти. Такие микросхемы естественно называть «суперкомпьютер-на кристалле» (supercomputer-on-chip). Например, отечественный процессор с шестнадцатью высокопроизводительными ядрами [2]. Технологии 7 нм позволяют на одной микросхеме располагать и более 1000 ядер [3, 4]. Такие же архитектуры могут быть сгенерированы на используемой в качестве ускорителя ПЛИС [5].

В работе [6] отмечено, что автоматическая компиляция последовательной программы для параллельной архитектуры с распределенной памятью является очень сложной задачей, для которой нет эффективного решения (на момент написания этой статьи). Тем не менее, в Южном федеральном университете ведутся работы по автоматическому распараллеливанию программ на вычислительные системы с распределенной памятью.

2. Особенности распараллеливающих компиляторов на вычислительные системы с распределенной памятью

Для вычислительных систем с распределенной памятью (ВСРП) самой длительной операцией является пересылка данных между процессорными элементами (ПЭ). Для высокопроизводительных кластеров такие пересылки могут даже вызвать замедление, поскольку их длительность в десятки раз больше времени выполнения вычислительных операций. Для эффективного отображения на ВСРП программа должна удовлетворять таким жестким требованиям, что ускорение может быть достигнуто для узкого класса программ, и разработка компилятора нецелесообразна. Но в последнее время появляются многоядерные микросхемы, называемые «суперкомпьютер на кристалле», с десятками, сотнями и даже тысячами ядер, большинство из которых ориентированы на реализацию нейронных сетей [2-4]. Пересылка данных между процессорными ядрами одной микросхемы требует значительно меньше времени, чем на коммуникационной сети кластера. Это означает расширение множества эффективно распараллеливаемых программ и делает целесообразным разработку распараллеливающих компиляторов.

В [7] отмечена перспективность циклических пересылок данных для микросхем близкого будущего (с тысячами ядер). В [8] приведено много задач линейной алгебры и математической физики, для параллельного решения которых на ВСРП использованы циклические пересылки. Метод размещения данных с перекрытиями [2] существенно ускоряет параллельные итерационные алгоритмы, уменьшая количество пересылок при укрупнении множеств пересылаемых данных. В [9] описаны блочно-аффинные размещения данных в распределенной памяти.

В [10] рассмотрена задача распараллеливания программного цикла на ВСРП, где для минимизации межпроцессорных пересылок по тексту программы построен

вспомогательный граф «операторы-переменные» и приведены примеры, основанные на ОРС реализации [11].

Подход, представленный ниже, отличается от других попыток автоматизировать отображение последовательных программ на ВСРП тем, что оптимальные размещения массивов находятся среди блочно-аффинных размещений массивов [9, 10], которые, с одной стороны, могут быть описаны малым количеством параметров (пропорционально размерности массива), а, с другой стороны, покрывают размещения, используемые на практике, включая такие, как скошенная форма хранения матрицы [8]. Расширение множества распараллеливаемых программ может быть достигнуто с помощью оптимизирующих преобразований программ, которые имеются в ОРС (Оптимизирующая распараллеливающая система) и в известных оптимизирующих компиляторах LLVM, GCC, ICC, MS-compiler.

Можно выделить следующие специфичные для ВСРП функции и преобразования:

- Размещение данных в распределенной памяти.
- Поиск оптимального множества циклических пересылок с помощью ГОП (граф «операторы-переменные»).
- Транспонирование матриц, размещенных в распределенной памяти, и его обобщение на многомерные массивы.
- Размещение массивов данных кратными с перекрытиями.
- Группировка пересылок.
- Определение оптимального количества ПЭ.
- Оптимизация преобразований многомерных массивов: транспонирование и скашивание.

К преобразованиям, специфическим для ВСРП, можно отнести распознавание и вызов пересылок данных для многомерных массивов, таких как транспонирование и скашивание.

В [12, 13] показано, что такие преобразования проще разрабатывать на основе высокоуровневого внутреннего представления программ. Такое внутреннее представление есть в компилирующих системах ROSE-compiler и ОРС. Высокоуровневое внутренне представление есть также системах SUIF, PLUTO и PolyLLVM. Однако SUIF более 20 лет назад перестала развиваться, а полиэдральное внутреннее представление PLUTO и PolyLLVM имеет узкую направленность на преобразования гнезд циклов.

В ОРС есть визуализация преобразований и информационных зависимостей, которая позволяет быстрее разработчику компилятора познакомиться с соответствующими преобразованиями программ. Такой визуализации нет других компиляторах.

Граф информационных связей (ГИС) в ОРС в качестве вершин имеет вхождения переменных. В промышленных оптимизирующих компиляторах для анализа информационных зависимостей используется граф зависимостей по данным (data dependence graph), у которого вершинами являются операторы. Эти два графа связаны следующим образом: граф зависимостей по данным является факторграфом графа информационных связей, если в графе зависимостей по данным склеивать все вершины, принадлежащие одному и тому же оператору. Для отображения программ на нестандартные архитектуры (конвейерные, с распределенной памятью) удобнее граф информационных связей. Для конвейерных архитектур ГИС удобнее, поскольку в схеме генерируемого конвейера вхождениям переменных соответствуют операции чтения (записи) данных. ГИС является более удобным для оптимизации программ для систем с распределенной памятью, поскольку используемый при минимизации

пересылок граф «операторы-переменные» (см. [10]) имеет вхождения переменных в качестве ребер.

3. О взаимодействии проектируемой системы высокоуровневых преобразований к LLVM.

Различных ВСРП может быть больше, чем различных систем с общей памятью. Поэтому фраза «распараллеливающий компилятор на ВСРП» должна пониматься, как некий инструмент для создания распараллеливающих компиляторов на различные ВСРП.

Многие компании – разработчики процессоров, делают компиляторы, основанные на LLVM. В системе LLVM кроме библиотеки преобразования программ есть много генераторов кода на различные целевые процессоры и парсеры с многих языков. Поэтому взаимодействие разрабатываемого проекта СПП с LLVM очень интересно.

Предполагается создание конвертора из LLVM в ОРС (например, через язык С), в результате чего объединятся библиотеки преобразований программ LLVM (https://llvm.org/docs/Passes.html) и ОРС. После создания такого конвертора, создаваемая СПП может рассматриваться как фреймворк LLVM. Оптимизирующие преобразования и функции СПП смогут применяться к фрагментам программ других языков, с которых есть парсер в LLVM. С другой стороны, в СПП можно будет использовать хорошо отлаженную библиотеку преобразований LLVM.

В LLVM реализована многопроходная стратегия оптимизации программ. Эта стратегия недостаточно хорошо оптимизирует программы [14]. В СПП предполагается поиск оптимальных цепочек преобразований. Поиск оптимальной цепочки преобразований – это сложная задача, которая не рассматривается современными компиляторами, которые заменяют решение этой задачи серией нескольких проходов (passes). Поиск оптимальной цепочки преобразований включает в себя поиск фрагментов кода – кандидатов на оптимизацию. Такая стратегия оптимизации требует большего времени для компилятора, чем многопроходная. Но это оправдано тем, что на современных процессорах компиляция, как и другие программы, выполняется быстрее. Для поиска оптимальной цепочки преобразований можно использовать распараллеливание.

4. Заключение

В настоящей работе предложен проект системы преобразования программ для создания распараллеливающих компиляторов на современные вычислительные системы, в том числе, с распределенной памятью.

Существует много приложений, для которых производительность существующих вычислительных систем недостаточна: прогноз погоды и изменений климата, отслеживание угроз столкновения метеоритов с Землей, планирование экономики, создание новых лекарств, решение задач метагеномики и др. Потребность в новых высокопроизводительных программно-аппаратных комплексах будет у общества еще многие годы. Поэтому реализация проекта системы преобразования программ вполне целесообразна.

Список литературы

1. Абу-Халил Ж.М., Гуда С.А., Штейнберг Б.Я., Перенос параллельных программ с сохранением эффективности // Открытые системы. СУБД, 2015, № 4. С. 18-19.]
2. Процессор НТЦ «Модуль». https://www.cnews.ru/news/top/2019-03-06_svet_uvidel_moshchnejshij_rossijskij_nejroprotsessor (дата обр. 26.03.2022).
3. SoC Esperanto. https://www.esperanto.ai/technology/
4. Peckham O. SambaNova. Launches Second-Gen DataScale System. https://www.hpcwire.com/2022/09/14/sambanova-launches-second-gen-datascalesystem/.
5. Dordopulo A.I., Levin I.I., Gudkov V.A., Gulenok A.A. High-Level Synthesis of Scalable Solutions from C-Programs for Reconfigurable Computer Systems // Malyshkin V. (Ed.) Parallel Computing Technologies. PaCT 2021. Lecture Notes in Computer Science. Cham: Springer, 2021. Vol. 12942. https://doi.org/10.1007/978-3-030-86359-3_7.
6. Bondhugula U. Automatic distributed-memory parallelization and codegeneration using the polyhedral framework. Technical report, ISc-CSA-TR-2011-3, 2011. 10 p.
7. Корнеев В.В. Параллельное программирование // Программная инженерия. 2022. Т. 13, № 1. С. 3-16.
8. Прангишвили И.В., Виленкин С.Я., Медведев И.Л. Параллельные вычислительные системы с общим управлением. М.: Энергоатомиздат, 1983. 312 с.
9. Штейнберг Б.Я. Блочно-аффинные размещения данных в параллельной памяти // Информационные технологии. 2010. № 6. С. 36-41.
10. Krivosheev N.M., Steinberg B.Ya. / Algorithm for searching minimum inter-node data transfers. // 10th International Young Scientist Conference on Computational Science YSC 2021. 1-3 July 2021. P. 306-313.
11. Оптимизирующая распараллеливающая система www.ops.rsu.ru.
12. Bagliy A.P., Krivosheev N.M., Steinberg B.Ya. Automatic mapping of sequential programs to parallel computers with distributed memory // 2023. Vol. 229. P. 236-244.
13. Штейнберг Б.Я. О создании распараллеливающих компиляторов на вычислительные системы с распределенной памятью // Труды Всероссийской научной конференции «Научный сервис в сети Интернет: технологии распределенных вычислений». 18-22 сентября 2023 г., он-лайн., ИПМ им. Келдыша, Москва. 11 с. https://keldysh.ru/abrau/2023/temp/23.pdf.
14. Gong Z., Chen Z., Szaday Z., Wong D., Sura Z., Watkinson N., Maleki S., Padua D., Veidenbaum A., Nicolau A. An empirical study of the effect of source-level loop transformations on compiler stability // Proceedings of the ACM on Programming Languages. 2018. P. 1-29.