ИССЛЕДОВАНИЕ РЕАЛИЗАЦИИ АЛГОРИТМА SURVEY PROPAGATION ДЛЯ РЕШЕНИЯ ЗАДАЧИ ВЫПОЛНИМОСТИ ФУНКЦИЙ БУЛЕВЫХ ПЕРЕМЕННЫХ (SAT-ЗАДАЧА) НА ЯЗЫКЕ UPC

Д.В. Андрюшин, А.С. Семенов

1. Язык UPC

Язык программирования Unified Parallel C (UPC) [1] является расширением языка ANSI C (ISO 99) и относится к классу PGAS-языков (Partitioned Global Address Space). Программная модель UPC относится к классу SPMD, множество процессов (тредов) выполняют один и тот же программный код. В UPC поддерживается глобальное адресное пространство, разделенное на части, количество частей равно количеству тредов, из которых состоит программа. Между тредами и частями глобального адресного пространства существует взаимнооднозначная связь, при этом обращение треда к “своей” части глобального пространства происходит намного быстрее, чем обращение к другой части. При этом для доступа к удаленным данным не требуются какие-либо вспомогательные операции копирования или приема/посылки сообщений. Компилятор генерирует соответствующие операции автоматически, создавая видимость программирования в едином адресном пространстве.

В последнее время язык UPC получает все большее распространение, помимо поддержки UPC на заказных суперкомпьютерах Cray X1/X1E, Cray XT3/4/5, Cray XE6, IBM Blue Gene L/P, системах фирмы SGI, существуют реализации UPC для кластеров на основе коммерчески доступных компонентов. Для исследования применимости UPC на таких кластерах была взята задача проверки выполнимости функций булевых переменных, обладающая интенсивным нерегулярным доступом к памяти и являющаяся сложной для эффективного решения на суперкомпьютерах. В исследовании использовалась открытая реализация Berkeley UPC [2].

2. Алгоритм Distributed Survey Propagation

Задача проверки выполнимости и поиска выполняющего набора для функции в КНФ (SAT-задача, Satisfiability problem) является NP-полной. К SAT-задачам сводятся многие практически важные проблемы: задача синтеза и верификации дискретных управляющих схем, задачи теоретического программирования, вычислительной биологии, а также многочисленные криптографические задачи [3].

Задача выполнимости (SAT-задача) формулируется следующим образом: существует ли набор булевых значений переменных, входящих в формулу в КНФ (конъюнктивной нормальной форме), при которых формула принимает значение *true = 1*. Если такой набор значений переменных существует, то его называют выполняющим, а формулу выполнимой.

Мы будем рассматривать случайную K-SAT задачу при K=3, в которой присутствует формула в КНФ (конъюнктивной нормальной форме) от N – переменных с $M=\alpha N$ дизъюнктов, причем каждый из дизъюнктов содержит в точности K – переменных. Формула в КНФ является случайной: номера переменных в каждом дизъюнкте случайны, дизъюнкты не повторяются, вхождение переменной в дизъюнкт с отрицательным или положительным знаком равновероятно.

В исследовании использован алгоритм distributed survey propagation (DSP), описанный в работах [4, 5].

Кратко суть алгоритма состоит в следующем. Каждой формуле в КНФ можно поставить в соответствие граф, состоящий из двух типов вершин: первый тип соответствует дизъюнктам формулы, второй – переменным формулы. Вершина-дизъюнкт соединена с вершиной-переменной ребром, если данная переменная входит в дизъюнкт.

Каждый узел-дизъюнкт *a* формирует сообщение узлу-переменной *i* в виде действительного числа. Сообщения формируются на основании предсообщений с предыдущей итерации или выбираются случайными в начале работы. Каждый узел-переменная, на основании всех принятых сообщений от узлов-дизъюнктов, формирует предсообщения (действительные числа) узлам-дизъюнктам. Сообщения характеризуют вероятность того, что переменная выполняет дизъюнкт и того, что дизъюнкт для выполнения нуждается в данной переменной.

На основании сообщений от узлов-дизъюнктов для узлов-переменных выполняется расчет «поляризации» переменной. Для каждого дизъюнкта на основании поляризаций переменных, входящих в дизъюнкт, вычисляется поляризация дизъюнкта. Путем сравнения поляризации переменной с поляризациями дизъюнктов, в которые она входит, принимается решение о фиксировании переменной и присваивании ей определенного значения. Далее алгоритм повторяется до тех пор, пока максимальная поляризация всех переменных и дизъюнктов не будет меньше заданного порога или не пройдет заданное количество итераций.

3. Реализация

Алгоритм survey propagation сложен для эффективной реализации, так как постоянно требуется обращаться к большому количеству нерегулярных структур данных, причем разными способами. Известно, что этот алгоритм эффективно работает на суперкомпьютере на основе мультитредового процессора Cray XMT [6].

Первый вариант параллельной программы получен на основе базового последовательного варианта на языке C при помощи небольших изменений. Изменения касаются распределения данных и разделения работы по тредам. Треды разделяют между собой часть массивов по переменным, часть – по дизъюнктам.

Работа разделяется по тредам в соответствии с распределением данных, в первой фазе алгоритма каждый тред обрабатывает свои переменные, считывая при этом поляризации дизъюнктов из удаленной памяти и записывая в удаленную память полученные предсообщения. На второй фазе алгоритма каждый тред обрабатывает свои дизъюнкты, считывая из глобальной памяти поляризации переменных, и записывает в нее посчитанные сообщения.

Такой вариант программы на UPC показывает крайне низкие результаты как по времени работы, так по масштабируемости. Основные причины низкой производительности заключается в следующем:

- Велики накладные расходы при работе с указателями на глобально адресуемые данные (pointer-to-shared), даже если в результате происходит обращение к локальной памяти треда.
- Происходит большое количество записей в удаленную память рассчитанных значений сообщений и рассчитанных значений предсообщений, а также чтений из удаленной памяти данных о поляризации переменных и дизъюнктов. Записи и чтения происходят по случайным индексам поэлементно.

Во втором варианте параллельной программы на UPC удалось избавиться от накладных расходов, связанных с пунктом 1. В первом варианте программы вся работа с массивами велась по указателям на глобально адресуемые данные. Средства UPC позволяют преобразовывать такие указатели в указатели на локальные данные треда, что и было сделано во втором варианте. Таким образом, указатели на глобально адресуемые данные используются только для обращений к удаленной памяти. В табл. 1. представлено сравнение производительности базового варианта на языке C и вариантов на языке UPC при последовательном выполнении на одном треде. Результаты показывают, насколько велики накладные расходы при расточительной работе с указателями UPC pointer-to-shared.

В результате выполненных действий у реализации появилась масштабируемость, однако абсолютные показатели времени работы и производительности на нескольких узлах оставались неудовлетворительными.

Табл 1. Сравнение производительности (Mflops) различных вариантов реализации DSP-алгоритма при выполнении одного треда на одном узле, N = 32000, α =4.21.

Последовательный вариант (C)	**Вариант 1 (UPC)**	**Вариант 2 (UPC)**	**Вариант 3 (UPC)**
67.33	15.33	59.93	51.27

Из-за низких результатов был разработан третий вариант. В третьем варианте поэлементная работа с массивами в удаленной памяти, влекущая использование коротких сетевых пакетов, заменена на работу с длинными сообщениями, которые поддерживает UPC при помощи функций upc_memput и upc_memget.

Для использования upc_memput вместо массивов для хранения предсообщений и сообщений заведены специальные структуры данных в памяти каждого треда. Перед началом счета предсообщений тред получает от других тредов список всех сообщений (вместе с поляризацией дизъюнктов) для своих переменных. Перед началом счета сообщений тред получает список всех предсообщений (вместе с поляризацией переменных) для своих дизъюнктов от всех тредов.

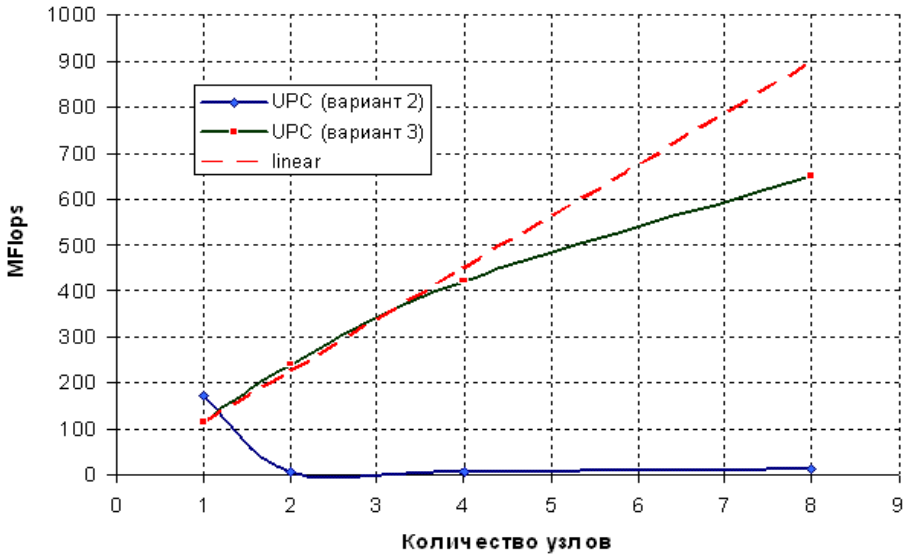

Рис. 1. Производительность второго и третьего вариантов программы на UPC в зависимости от количества используемых узлов, N=32000, =4.21.

Эксперименты производились на 8-ми узловом кластере с двухпроцессорными узлами на основе двухядерного Opteron 285 2.6 ГГЦ, коммуникационная сеть – Infiniband 4xDDR. На рисунке 2 показана производительность второго и третьего вариантов. Видно, что третий вариант показывает значительно лучшую производительность и масштабируемость.

4. Заключение

Полученные результаты показывают, что написание эффективных программ на языке UPC при выполнении на кластере на основе коммерчески доступных компонентов требует определенного навыка. Губительно для производительности свободно использовать указатели на глобально адресуемые данные. Также для UPC естественно использовать поэлементные пересылки, которые не в состоянии эффективно поддержать сеть Infiniband 4xDDR. В результате на UPC удалось получить эффективную реализацию, однако при этом использовались длинные сообщения.

Работа является незаконченной, в дальнейшем авторы планируют сравнение полученного варианта на UPC с программой, использующую для коммуникаций библиотеку MPI.

Результаты получены в рамках выполнения программы СКИФ-ГРИД. За постановку задачи авторы выражают благодарность Л. К. Эйсымонту.

ЛИТЕРАТУРА:

1. UPC Consortium, UPC Language Specification, v. 1.2 // Lawrence Berkeley National Lab Tech Report, 2005.
2. The Berkeley UPC Compiler, http://upc.lbl.gov/ .
3. J. Gu, P.W. Purdom, J. Franco, B.W. Wah, “Algorithms for Satisfiability (SAT) problem: a Survey“ // In DIMACS Series in Dicrete Maths. and TCS., AMS, Vol.35, pp. 19-152, 1996.
4. A. Braunstein, M. Mezard, R. Zecchina, “Survey Propagation: an algorithm for satisfiability” // Random Structures and Algorithms 27, pp. 201-226, 2005.
5. J. Chavas, C. Furtlehner, M. Mezard, R.Zecchina, “Survey Propagation decimation through distributed local computations” // J. Stat. Mech., 2005.
6. D. Chavarria-Miranda, D.K. Gracio, A. Marquez, J. Nieplocha, C. Scherrer, and H.J. Sofia, "Cray XMT Brings New Energy to High-Performance Computing." // SciDAC Review Fall (9), pp. 36-41, 2008.