РЕЗУЛЬТАТЫ СРАВНЕНИЯ РЕАЛИЗАЦИЙ UPC И CAF НА ТЕСТАХ NPB

М.В. Кудрявцев, А.В. Литвинович

Аннотация

В тезисах приведены результаты предварительного исследования реализаций языков UPC и CAF на тестах NPB на одном и двух узлах кластера (по четыре шестиядерных процессора AMD Opteron 8431 “Istanbul”, 2.4 ГГц на каждом) с сетью Infiniband 4X DDR. Для сравнения приведены результаты одноузловых запусков тестов NPB, реализованных на MPI и OpenMP.

Языки программирования, позволяющие пользователю разрабатывать программы в моделях с глобальным адресным пространством с неоднородным доступом (partitioned global address space) [1] набирают все большую популярность. В 2010 году в США будет проводиться уже четвертая специализированная конференция по моделям программирования PGAS [2]. Общая идея PGAS-моделей состоит в том, что данные в программе делятся на два типа - локальные и удаленные. К локальным данным доступ осуществляется быстрее. При этом для доступа к удаленным данным пользователю не требуется определять какие-либо вспомогательные операции копирования или посылки/приема сообщений, компилятор генерирует соответствующие обращения автоматически, создавая иллюзию программирования в глобальном адресном пространстве. К PGAS-языкам относят языки UPC [3], CAF [4], Titanium [5], X10 [6], Chapel [7] и др.

Программная модель UPC относится к классу SPMD, множество независимых легковесных процессов (тредов) выполняют один и тот же программный код. Память каждого треда разделена на две части: локальную и общую (shared). Глобальное адресное пространство состоит из общих сегментов памяти всех тредов. Количество сегментов равно количеству тредов, из которых состоит программа. Между тредами и сегментами существует взаимнооднозначная связь, при этом обращение к памяти, находящейся в соответствующем треду сегменте происходит намного быстрее, чем обращение к другому сегменту общей памяти. Доступ к общей памяти осуществляется через использование общих объектов. Общими объектами могут быть переменные, массивы и динамически выделенные сегменты памяти. Распределение массивов происходит блочно-циклически по принципу round-robin. Для каждого общего массива указывается размер блока разбиения. Как размер общего массива, так и размер блока разбиения могут задаваться динамически, через количество тредов. Распределение вычислений определяется программистом либо явно через операторы условного перехода, либо неявно через распределение общих данных и установление соответствия между этими данными и вычислениями. Программы на UPC могут быть написаны с использованием различных моделей консистентности.

Программная модель CAF является небольшим синтаксическим расширением языка Fortran 95/2003 и включена в стандарт 2008 года. Эта модель тоже относится к классу SPMD – программа состоит из множества копий, в терминах CAF – образов (images), асинхронно выполняющихся на процессорах. Количество образов может равняться числу используемых процессоров, а может превосходить его. Каждый из образов обладает своим собственным набором данных, часть которых может быть доступна из других образов. Для доступа к данным других образов в CAF введено понятие общего массива (co-array), обеспечивающего распределение данных по образам. Синтаксис определения общих массивов и доступа к их элементам во многом схож с синтаксисом определения обычных массивов. При обращении к элементу общего массива индекс в квадратных скобках определяет образ, где находится элемент, а индекс в круглых - позицию элемента в локальной памяти этого образа. Весь обмен данными между образами осуществляется посредством операций чтения/записи в общие массивы. Оба языка CAF и UPC имеют средства синхронизации образов (нитей).

Существует несколько реализаций языка UPC, из них регулярно поддерживается и доступна для загрузки и использования на большом количестве платформ, включая кластера на основе Infiniband, реализация университета Беркли, которая и использовалась в настоящем исследовании. Для CAF-программ использовался компилятор и система времени выполнения CAF 1.0, разработанные в университете Райз.

Группами университетов Беркли и Райз на языки UPC и CAF были перенесены некоторые тесты NPB (NASA performance benchmarks) [8], доступные в оригинале на языках Си и Фортран и параллельных расширениях MPI, OpenMP. Пакет NPB состоит из пяти параллельных ядер и трех модельных приложений, имитирующих вычисления и характеристики перемещения данных больших приложений гидрогазодинамики. Настройки пакета позволяют регулировать объем обрабатываемых тестами данных, который для каждого теста индивидуален и обозначается буквами латинского алфавита.

Существует множество исследований качественных характеристик языков параллельного программирования и их систем времени выполнения на тестах пакета NPB [9,10]. Мы решили провести подобное исследование на одном и двух узлах современного кластера с мощными SMP-узлами (24 ядра на узел), воспользовавшись открытыми реализациями языков UPC и CAF и сравнив производительность тестов на них с результатами на MPI и OpenMP. На следующих рисунках приведены графики производительности программ NPB класса C (больший объем данных) и B в пересчете на одно ядро. Данные приведены для всех доступных реализаций тестов NPB, проходящих встроенную проверку корректности результата.

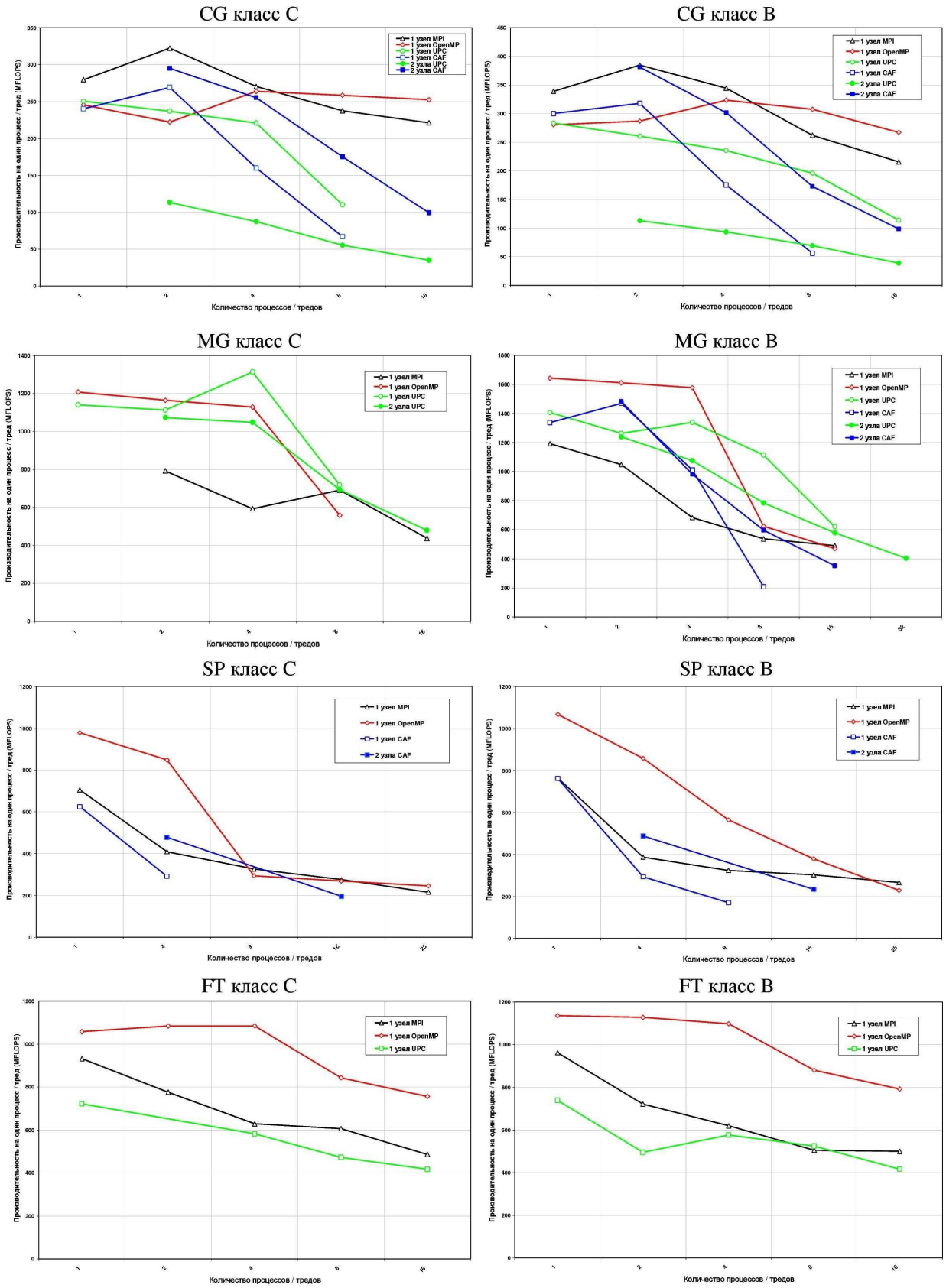

Рис. 1

В исследовании использовались компиляторы Intel 10.0 (Фортран) и 10.1 (C) с поддержкой OpenMP и библиотека MPI mvapich 1.1.0. Все программы компилировались с оптимизацией (-O3). Узлы соединены сетью Infiniband 4x DDR без коммутатора двумя портами. Объем доступной на узле оперативной памяти - 16 Гб. Использовались CAF и UPC-версии тестов на основе пакета NPB 2.3, MPI и OpenMP-версии - из пакета NPB

3.3. На представленных рисунках в некоторых точках отсутствуют данные. Это связано либо с нехваткой оперативной памяти на узле, либо с некорректным результатом теста.

UPC и CAF версии теста CG обгоняют по производительности OpenMP версию только на 2 процессорах, проигрывая при этом реализации на MPI. UPC-тест CG сильно деградирует по производительности при переходе на 2 узла, в отличие от теста на CAF, который заметно ускоряется. MPI-реализация теста CG оказывается эффективнее всех остальных, UPC и CAF версии сильно деградируют в производительности при увеличении количества процессоров.

Производительность на тесте MG имеет меньший разброс. Пожалуй, наиболее интересно то, что OpenMP-версия сильно опережает MPI на небольшом количестве процессоров. UPC-версия тоже опережает MPI, а в некоторых местах даже и OpenMP-версию. CAF-версия опережает MPI на небольшом количестве процессоров.

Что касается теста SP, то результаты CAF-версии близки к результатам на MPI. При этом OpenMP-версия опережает MPI и CAF, это особенно заметно на небольшом количестве процессоров.

На тесте FT производительность UPC-версии близка к MPI, в то время как OpenMP-версия сильно опережает MPI.

Полученные результаты свидетельствуют о том, что используя существующие открытые реализации языков UPC и CAF можно получить производительность, близкую к MPI, а в некоторых случаях и превышающую OpenMP. Проигрыш в производительности в некоторых случаях является недостатком текущих реализаций языков, а не самих моделей программирования UPC и CAF, которые очень хорошо отражают возможности аппаратуры современных кластеров (RDMA-операции Infiniband, многоядерность и мультитредовость). По нашему мнению, стоит ожидать дальнейшее совершенствование реализаций этих языков и их постепенное внедрение в промышленности.

В дальнейшем планируется провести исследование причин недостаточно высокой производительности UPC и CAF-программ, а также провести измерения на простых тестах, таких как STREAM [11].

Авторы выражают признательность ЗАО «Троник» за предоставленное для исследования оборудование.

Результаты получены в рамках выполнения программы СКИФ-ГРИД.

ЛИТЕРАТУРА:

1. http://en.wikipedia.org/wiki/Partitioned_global_address_space
2. http://groups.google.com/group/pgas10/web/call-for-papers
3. http://upc.lbl.gov
4. http://caf.rice.edu/index.html
5. http://titanium.cs.berkeley.edu
6. http://x10-lang.org
7. http://chapel.cray.com
8. http://www.nas.nasa.gov/Resources/Software/npb.html
9. C. Coarfa, Y. Dotsenko, J. Mellor-Crummey, F. Cantonnet, T. El-Ghazawi, A. Mohanti, Y. Yao, D. Chavarría-Miranda, 2005. An evaluation of global address space languages: co-array fortran and unified parallel C. In *Proceedings of the Tenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (Chicago, IL, USA, June 15 - 17, 2005). PPoPP '05. ACM, New York, NY, 36-47. DOI= http://doi.acm.org/10.1145/1065944.1065950
10. В.А. Крюков. Разработка параллельных программ для вычислительных кластеров и сетей. Проблемы и перспективы. Тезисы докладов Третьей Всероссийской молодежной школы "Суперкомпьютерные вычислительно-информационные технологии в физических и химических исследованиях". г. Черноголовка, 31 октября - 1 ноября 2001г. Изд-во ИПХФ РАН стр. 106-121
11. http://www.cs.virginia.edu/stream