# ИСПОЛЬЗОВАНИЕ ЯЗЫКА FORTRAN-DVM/OPENMP ДЛЯ РЕШЕНИЯ БОЛЬШИХ ЗАДАЧ

**В.А. Бахтин, А.В. Воронков, А.С. Голубев, В.А. Крюков, Н.В. Поддерюгина, Е.П. Сычугова, С.Д. Устюгов**

## ВВЕДЕНИЕ

Последние годы связаны с резким изменением направления развития процессоров – появлением многоядерных и многопоточных процессоров. Их эффективное использование требует повсеместного перехода с последовательных программ на параллельные программы. Для высокопроизводительных вычислений на кластерах и многопроцессорных ЭВМ с массовым параллелизмом (MPP) теперь требуется более глубокое распараллеливание, обеспечивающее, по крайней мере, два уровня параллелизма – уже привычный параллелизм между узлами кластера и дополнительный параллелизм между ядрами в узле. В настоящее время такой двухуровневый параллелизм наиболее легко и естественно выражать посредством использования между узлами модели передачи сообщений MPI[1], а внутри узлов – модели общей памяти Pthreads, или более высокоуровневой и более подходящей для вычислительных программ модели OpenMP[2]. Однако, необходимость более глубокого распараллеливания и использования гибридной модели MPI/OpenMP, т.е. двух разных моделей программирования и соответствующих им разных инструментальных средств еще более усложняет и без того нелегкую работу по созданию параллельных программ. Поэтому проблема автоматизации создания параллельных программ становится в настоящее время чрезвычайно актуальной.

Разработанная в ИПМ им. М.В.Келдыша РАН гибридная модель (DVM/OpenMP-модель) параллельного программирования и язык Fortran-DVM/OpenMP, позволяет существенно автоматизировать разработку программ для SMP-кластеров (кластеров, использующих в качестве узлов мультипроцессоры).

## ГИБРИДНАЯ МОДЕЛЬ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ DVM/OPENMP. ЯЗЫК FORTRAN-DVM/OPENMP

Рассмотрим вычислительную сеть, каждый узел которой является мультипроцессором или многоядерным процессором (**рис. 1**). На каждом узле выполняется параллельная программа на языке Fortran OpenMP. Распределение данных и вычислений между узлами, а также доступ к удаленным данным выполняется согласно модели DVM[3].

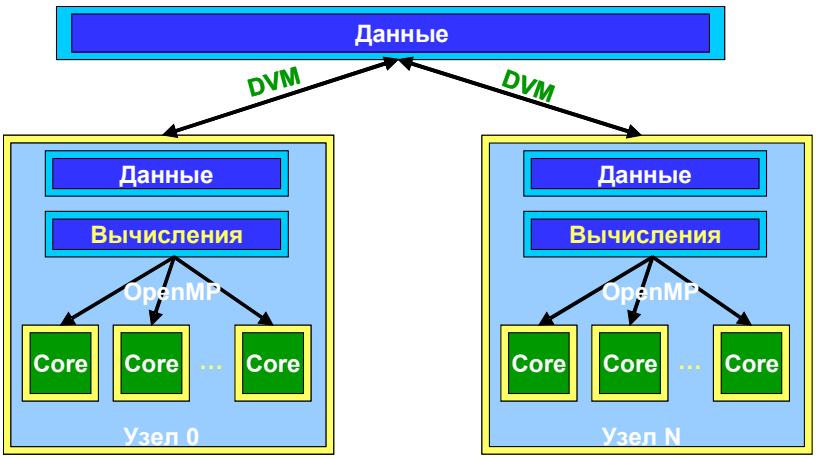


Рис. 1. Модель DVM/OpenMP

В программе на языке Fortran-DVM/OpenMP[4] можно описать следующие уровни параллелизма:

1. Параллелизм распределенных задач, позволяющий отобразить параллельно выполняющиеся задачи на непересекающиеся многомерные секции узлов.
2. Параллелизм распределенных многомерных циклов, витки которых распределяются между узлами многомерной секции.
3. Параллелизм конструкций разделения работы OpenMP (циклы и секции). Эти конструкции могут быть в произвольном порядке вложены друг в друга.

Любые из указанных уровней в программе могут отсутствовать. Если в программе описаны несколько уровней параллелизма, то порядок их вложенности должен соответствовать порядку их перечисления в приведенном выше списке.

Параллелизм распределенных задач реализуется распределением данных и независимых вычислений на секции массива узлов. Параллелизм распределенных многомерных циклов реализуется распределением витков тесно-гнездового цикла между узлами. При этом каждый виток такого параллельного цикла полностью выполняется на одном узле.

Спецификации параллелизма языка Fortran-DVM/OpenMP оформляются как спецкомментарии, что позволяет использовать разработанную программу и как стандартную последовательную программу, и как стандартную параллельную OpenMP-программу.

Рассмотрим пример параллельной программы на языке Fortran-DVM/OpenMP алгоритма численного решения двухмерного уравнения Лапласа методом Якоби.

```
      PROGRAM JAC_DVM_OpenMP
      PARAMETER (L=8, ITMAX=20)
      REAL A(L,L), B(L,L)
CDVM$ DISTRIBUTE (BLOCK, BLOCK) :: A
CDVM$ ALIGN B(I,J) WITH A(I,J)
C     arrays A and B  with block distribution
      PRINT *,  '********** TEST_JACOBI **********'
!$OMP PARALLEL
      DO IT = 1, ITMAX
CDVM$   PARALLEL (J, I) ON A(I, J)
!$OMP   DO
        DO J = 2, L-1
!$OMP     PARALLEL DO
          DO  I  =  2, L-1
            A(I, J)  =  B(I, J)
          ENDDO
        ENDDO
CDVM$   PARALLEL (J, I) ON B(I, J),SHADOW_RENEW (A)
C       Copying shadow elements of array A from
C       neighboring nodes before loop execution
!$OMP   DO
        DO J = 2, L-1
!$OMP     PARALLEL DO
          DO I = 2, L-1
            B(I, J) =  (A( I-1, J ) + A( I, J-1 ) +
     *            A( I+1, J ) + A( I, J+1 )) / 4
          ENDDO
        ENDDO
      ENDDO
!$OMP END PARALLEL
      END
```

При помощи директивы DISTRIBUTE массив A распределяется между узлами. Оператор вида A(I, J) = B(I, J), выполняемый в цикле, определяет требования по совместному распределению этих массивов. Чтобы исключить взаимодействие между узлами при выполнении этого оператора, необходимо распределить A(I,J) и B(I,J) элементы массива на один и тот же узел – для этого используется директива ALIGN. Директива PARALLEL ON каждый виток параллельного цикла ставит в соответствие некоторому элементу массива. Это означает, что (J,I) виток цикла будет выполняться на том узле, где распределен соответствующий элемент массива – A(I,J). Во втором цикле для вычисления элемента B(I,J) используются элементы A(I-1,J), A(I,J-1), A(I+1,J) и A(I,J+1). Поскольку массив A распределен, то эти элементы могут находиться на соседних узлах, поэтому перед выполнением цикла необходимо загрузить эти данные – для этого используется спецификация SHADOW_RENEW.

В результате выполнения директивы PARALLEL ON вычисления распределены по узлам. Для распределения этих вычислений внутри узла используются OpenMP-директивы. При помощи директивы PARALLEL создается группа нитей, между которыми при помощи директивы DO и распределяется вся работа (распределяются витки циклов, которые обрабатывают локальную часть массива).

Использование гибридной модели DVM/OpenMP позволяет упростить программирование в том случае, когда в программе есть два уровня параллелизма – параллелизм между подзадачами и параллелизм внутри подзадачи. Такая ситуация возникает, например, при использовании многообластных (многоблочных) методов решения вычислительных задач. Подзадачи программируются на OpenMP, а взаимодействие задач – на DVM. Программировать на DVM сами подзадачи гораздо сложнее, чем их взаимодействие, поскольку распараллеливание подзадачи связано с распределением элементов массивов и витков циклов между про-

цессами. Организация же взаимодействия подзадач таких сложностей не вызывает, поскольку сводится к обмену между ними граничными значениями. Нечто подобное программисты делали раньше на однопроцессорных ЭВМ, когда для экономии памяти на каждом временном шаге выполняли подзадачи последовательно друг за другом.

Использование OpenMP на узле SMP-кластера позволяет ликвидировать или сократить дублирование данных, используемых разными потоками. При использовании DVM, в каждом процессе могут дублироваться следующие данные:

- нераспределяемые (размноженные) при помощи директив DISTRIBUTE и ALIGN пользовательские массивы (например, массивы констант);
- заводимые системой поддержки выполнения DVM-программ (библиотекой Lib-DVM) буфера для обмена данными между процессами (например, теневые грани).

Все эти дополнительной затраты памяти становятся очень критичными для многоядерных узлов, поскольку имеется явная тенденция к сокращению объема оперативной памяти, приходящейся на одно ядро.

## РАСПАРАЛЛЕЛИВАНИЕ ТЕСТОВ NAS В МОДЕЛИ DVM/OPENMP

Тесты NAS (Numeric Aerodynamic Sumulation) широко используются для оценки и тестирования возможностей параллельных компьютеров, а также инструментов для разработки параллельных программ. При создании DVM/OpenMP-версий программ за основу были использованы MutliZone версии [5] тестов NAS, написанных в гибридной модели MPI/OpenMP.

В модели MPI/OpenMP реализованы три модельные приложения:

1. BT (Block Tridiagonal Solver) – неявный алгоритм нахождения конечно-разностного решения 3-х мерной системы уравнений Навье-Стокса для сжимаемой жидкости или газа. Используется блок-диагональная схема, метод переменных направлений.
2. LU (Lower-Upper Solver) – нахождение конечно-разностного решения 3-х мерной системы уравнений Навье-Стокса для сжимаемой жидкости или газа. Применяется метод LU-разложения, с использованием алгоритма SSOR (метод верхней релаксации).
3. SP (Scalar Pentadiagonal Solver) – неявный алгоритм нахождения конечно-разностного решения 3-х мерной системы уравнений Навье-Стокса для сжимаемой жидкости или газа. Используется скалярная пентадиагональная схема.

Для реализации MultiZone версий этих приложений использовался OVERFLOW-подход. Внутри счетной области выделяются подобласти – зоны. Каждая зона рассчитывается независимо от других. После выполнения каждой итерации, зоны обмениваются граничными значениями со своими ближайшими соседями. Для распределения зон по узлам вычислительного кластера и для обмена граничными значениями используется библиотека MPI. Для распределения вычислений внутри зоны используется OpenMP.

При создании версий этих программ в модели DVM/OpenMP, работа по отображению зон по узлам и взаимодействию между узлами была переписана на DVM с использованием механизма задач. OpenMP-директивы остались без каких-либо изменений.

Проведенное на машине Regatta (IBM eServer pSeries 690), установленной в Московском государственном университете им. М. В. Ломоносова, исследование характеристик эффективности выполнения программ показало, что гибридные модели DVM/OpenMP и MPI/OpenMP близки по эффективности. На одних и тех же конфигурациях (<число MPI-процессов>x<число нитей>) DVM и MPI-версии тестов ведут себя практически одинаково.

Для теста BT реализация на OpenMP дополнительного уровня параллелизма (что очень сложно было бы реализовать на MPI) позволила обеспечить лучшую балансировку загрузки процессоров, что привело к заметному ускорению программы (**рис. 2-5**).

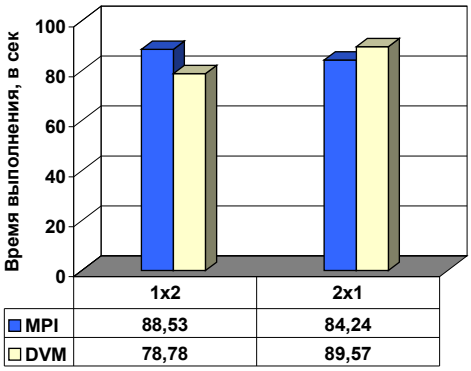


Рисунок 2. Время выполнения теста BT на 2-х процессорах

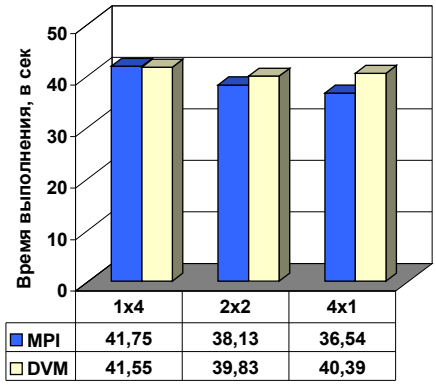


Рисунок 3. Время выполнения теста BT на 4-х процессорах

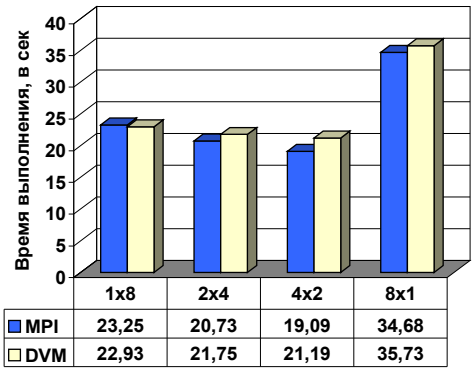


Рисунок 4. Время выполнения теста BT на 8-ми процессорах

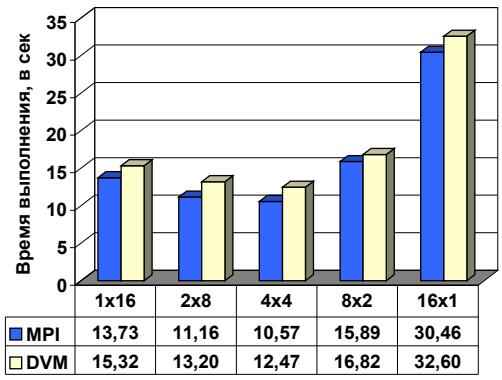


Рисунок 5. Время выполнения теста BT на 16-ти процессорах

В тесте BT – все зоны различного размера: от 13x13x16 и до 58x58x16 (для класса A). Поэтому зоны между MPI-процессами распределяются таким образом, чтобы обеспечить их одинаковую загрузку. При увеличении числа MPI-процессов свыше 4-х такая балансировка становится невозможной, поскольку количество зон ограничено (всего 16 зон для класса A). В результате, MPI-процессы фактически начинают работать со скоростью обработки самой медленной зоны. Вычислив свою зону (зоны), им приходиться ожидать пока другие процессы не посчитают свои зоны, для того чтобы затем обменяться граничными значениями. Увеличение числа MPI-процессов не приводит к дальнейшему ускорению программы. Использование OpenMP дает возможность сократить число MPI-процессов и улучшить балансировку. Минимальное время выполнения теста достигается при использовании 4-х MPI-процессов и 4-х нитей.

ЗАДАЧА CONVD – РАССЧЕТ СОЛНЕЧНОЙ КОНВЕКЦИИ В МОДЕЛИ DVM/OPENMP

Задача CONVD (задача С.Д. Устюгова)[6] считает трехмерную магнитную газодинамику для расчета солнечной конвекции с использованием реалистичной физической модели. Исследуется влияние магнитного поля на тепловую структуру конвективных перемещений в радиационном слое, характерные масштабы развития конвекции и глубина проникновения конвекции. Моделируется развитие конвекции в верхних слоях солнечной фотосферы на размерах локальной супергрануляции. Решаются уравнения полностью сжимаемой радиационной магнитной газодинамики с учетом динамической вязкости и гравитации.

Для численного моделирования используется:

- реальная первоначальная модель Солнца и уравнения состояния, непрозрачность звездной материи,
- консервативная TVD (Total Variation Diminishing) схема высокого порядка для решения магнитной газодинамики,
- диффузионное приближение для решения радиационного переноса,
- динамическая вязкость рассчитывается в зависимости от масштаба моделирования.

Для получения программы на языке Fortran-DVM/OpenMP использовалась DVM-версия этой программы. В следующей таблице (**таблица 1**) приведены размеры различных версий программы в строках.

| Последовательная программа | DVM-программа | DVM/OpenMP-программа |
|---|---|---|
| 3214 | 3380 | 3649 |

Таблица 1. Размер различных версий программы CONVD в строках

Существенное усложнение программы на языке Fortran-DVM/OpenMP (по сравнению с Fortran-DVM), прежде всего, связано с тем, что в задаче описаны циклы с зависимостью по данным, распараллеливание которых на OpenMP требует больших усилий (низкоуровневое распараллеливание).

Данная задача считается в Межведомственном суперкомпьютерном центре РАН на суперкомпьютере «МВС-50К». Задача очень требовательна к объему оперативной памяти. До проведения модернизации суперкомпьютера на вычислительном узле запускались 4-MPI процесса, которые могли использовать до 4 Гбайт оперативной памяти. DVM-версия задачи запускалась на 225 процессорах. После проведения модернизации (использования 4-х ядерных процессоров) – на узле стали запускаться 8-MPI процессов, а объем оперативной памяти остался прежним. В результате DVM-версия программы перестала работать (рабочее множество страниц виртуальной памяти не помещалось в оперативной памяти).

Многие пользователи суперкомпьютерного центра столкнулись с подобной проблемой. Поэтому в используемой там системе очередей была реализована возможность задания количества запускаемых на

узле MPI-процессов. Однако при уменьшении числа запускаемых на узле MPI-процессов возрастает количество требуемых задачей узлов – время ожидания задачи в очереди увеличивается, а предоставляемые ресурсы используются не полностью. Версия программы на языке Fortran-DVM/OpenMP использует все ресурсы на узле и поэтому выполняется заметно быстрее (**таблица 2**).

| 57 вычислительных узлов | Время выполнения, сек |
|---|---|
| DVM-программа<br>225 MPI-процессов (4 MPI-процесса на узле) | 259.50 |
| DVM/OpenMP-программа<br>225 MPI-процессов x 2 нити | 159.47 |

Таблица 2. Время выполнения задачи CONVD в МСЦ РАН

РАСПАРАЛЛЕЛИВАНИЕ В МОДЕЛИ DVM/OPENMP ПРОГРАММ ИЗ ПАКЕТА «РЕАКТОР»

Пакет прикладных программ «РЕАКТОР» [7] разработан в Институте прикладной математики им. М.В.Келдыша РАН и в течение многих лет эксплуатируется в ИПМ РАН и ряде других организаций. Пакет представляет собой совокупность большого числа независимых программ – функциональных модулей. Каждый такой модуль решает логически завершенный фрагмент общей задачи, например, ввод геометрической информации, подготовка групповых констант, расчет нейтронных полей, выгорание и т.д. Прикладной специалист для решения своей задачи должен сформировать и выполнить цепочку модулей, каждый из которых в качестве входных данных использует выходные данные предшествующих модулей.

В нестационарной задаче DF3D6T из пакета «Реактор» моделируется процесс пространственной кинетики ядерного реактора в диффузионном приближении. Задача DF3D6T предназначена для расчета поведения реактора в процессе вдвигания замедляющих стержней. Расчет производится на гексагональной конечноразностной сетке диффузионным методом. Для последовательности моментов моделируемого времени вычисляются значения полей нейтронов, значения источников нейтронов, а также выделяемая мощность.

Для создания версии этой программы на языке Fortran-DVM/OpenMP была использована DVM-версия программы. На следующей таблице (**таблице 3**) приведены размеры различных версий программы DF3D6T в строках.

| Последовательная программа | DVM-программа | DVM/OpenMP-программа |
|---|---|---|
| 17035 | 17152 | 17324 |

Таблица 3. Размер различных версий программы DF3D6T в строках

На следующей диаграмме (**рис. 6**) показаны времена выполнения программы в секундах в Межведомственном суперкомпьютерном центре РАН на машине «МВС-50К». Приводятся результаты, полученные на разном числе узлов. На узле запускалось 8 MPI-процессов или 4 MPI-процесса, а каждый процесс порождал две нити.

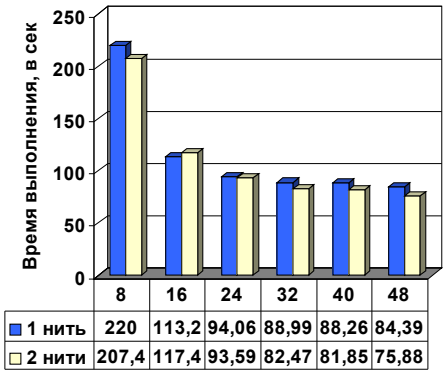


Рисунок 6. Времена выполнения программы DF3D6T на МВС-50К в МСЦ РАН

В рассматриваемой задаче применяется упрощенный метод распараллеливания по плоскостям, тестовые данные имели относительно небольшое число точек в горизонтальной плоскости (421), небольшое число плоскостей (96) и малое число групп (2). При увеличении числа используемых процессоров более 32, задача фактически перестает ускоряться, поскольку уменьшается работа, выполняемая одним процессором,

но при этом увеличивается время накладных расходов по взаимодействию процессоров. Существенного выигрыша от использования гибридной модели для данной задачи получено не было.

Задача KinXYZ используется для расчета нейтронных полей в кинетическом приближении в трехмерной X-Y-Z геометрии. Вычисляются поля нейтронов реактора для конструирования радиационной защиты.

Для создания версии этой программы на языке Fortran-DVM/OpenMP была использована DVM-версия программы. На следующей таблице (**таблица 4**) приведены размеры различных версий программы KinXYZ в строках.

| Последовательная программа | DVM-программа | DVM/OpenMP-программа |
|---|---|---|
| 14795 | 14897 | 14944 |

Таблица 4. Размер различных версий программы KinXYZ в строках

В данной задаче в каждом MPI-процессе приходится дублировать нейтронные макроконстанты, данные по запаздывающим нейтронам, данные по составу стержней. Для организации счета с продолжением, а также для анализа результатов счета используется файловый архив (бинарные файлы прямого доступа). Системные буфера, используемые системой поддержки Lib-DVM для организации ввода-вывода данных из архива, дублирование большого количества информации в памяти не позволяют использовать на одном вычислительном узле машины «МВС-50К» более 3-х MPI-процессов.

Применение гибридной модели для этой задачи позволило эффективнее использовать ядра узла. Вместо 3-х MPI-процессов на одном вычислительном узле запускается 8 нитей, что приводит более чем к 2-х кратному ускорению программы. На следующей диаграмме (**рис. 7**) приведено время выполнения 100 внутренних итераций, выполняемых для расчета одной области (группы), при использовании одного вычислительного узла.

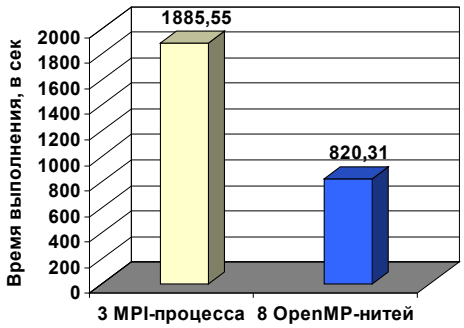


Рисунок 7. Времена выполнения программы KinXYZ на МВС-50К в МСЦ РАН при использовании одного узла

ЗАКЛЮЧЕНИЕ

Язык параллельного программирования Fortran-DVM/OpenMP позволяет существенно автоматизировать разработку программ для SMP-кластеров. Анализ приведенных в статье характеристик эффективности позволяет сделать следующие выводы:

- эффективность программ в гибридной модели DVM/OpenMP очень мало отличается от эффективности программ в гибридной модели MPI/OpenMP;
- эффективность программ в гибридной модели DVM/OpenMP не уступает эффективности программ в модели DVM;
- использование в узлах модели общей памяти OpenMP позволило получить следующие преимущества:
    - заметно ускорить решение реальных задач, требующих больших объемов оперативной памяти (за счет ликвидации дублирования информации в памяти, свойственного MPI-модели и DVM-модели);
    - заметно ускорить выполнение программы BT за счет реализации на OpenMP дополнительного уровня параллелизма, позволившей обеспечить лучшую балансировку загрузки процессоров.

Работа выполнялась в рамках научно-технических программ Союзного государства «ТРИАДА» и «СКИФ-ГРИД», а также Программы №14 Президиума РАН. Также была поддержана грантом Президента РФ для ведущих научных школ НШ-2139.2008.9 и грантом РФФИ № 08-07-00086.

ЛИТЕРАТУРА:

1. Message-Passing Interface Forum, Document for a Standard Message-Passing Interface, 1994. Version 1.0. (http://www.unix.mcs.anl.gov/mpi/)
2. OpenMP Consortium: OpenMP Application Program Interface, Version 3.0, May 2008. (http://www.openmp.org/mp-documents/spec30.pdf)
3. Fortran DVM – язык разработки мобильных параллельных программ / Н.А. Коновалов, В.А. Крюков, С.Н. Михайлов и др. // Программирование. – 1995. – № 1. – С. 49-54.
4. Описание языка Fortran-DVM/OpenMP. Версия 2.0, 2006 (http://www.keldysh.ru/dvm/omp/)
5. Rob F. Van der Wijngaart, Haoqiang Jin. NAS Parallel Benchmarks, Multi-Zone Versions. NAS Technical Report NAS-03-010. July 2003 (http://www.nas.nasa.gov/News/Techreports/2003/PDF/nas-03-010.pdf)
6. S.D. Ustyugov. Three Dimensional Numerical Simulations of Near Surface Solar Convection with Realistic Physics. Proceedings of the SOHO 14 / GONG 2004 Workshop (ESA SP-559). “Helio – and Asteroseismology: Towards a Golden Future”. 12-16 July, 2004. New Haven, Connecticut, USA.
7. A. Voronkov, V. Arzhanov. REACTOR – Program System for Neutron-Physical Calculations. Proc. International Topical Meeting: Advances in Mathematics, Computations, and Reactor Physics, USA, Vol5, April 28 – May 2, 1991