



В.А. Бахтин, Д.А. Захаров, А.Н. Козлов,
В.С. Коновалов

**Разработка параллельного
программного кода для расчетов
задачи радиационной магнитной
газодинамики и исследования
динамики плазмы в канале КСПУ**

Рекомендуемая форма библиографической ссылки

Бахтин В.А., Захаров Д.А., Козлов А.Н., Коновалов В.С. Разработка параллельного программного кода для расчетов задачи радиационной магнитной газодинамики и исследования динамики плазмы в канале КСПУ // Научный сервис в сети Интернет: труды XXI Всероссийской научной конференции (23-28 сентября 2019 г., г. Новороссийск). — М.: ИПМ им. М.В.Келдыша, 2019. — С. 105-118. — URL: <http://keldysh.ru/abrau/2019/theses/80.pdf>
doi:[10.20948/abrau-2019-80](https://doi.org/10.20948/abrau-2019-80)

Размещена также [презентация к докладу](#)

Разработка параллельного программного кода для расчетов задачи радиационной магнитной газодинамики и исследования динамики плазмы в канале КСПУ

В.А. Бахтин^{1,2}, Д.А. Захаров¹, А.Н. Козлов^{1,2}, В.С. Коновалов¹

¹ *ИИМ им. М.В. Келдыша РАН*

² *МГУ им. М.В. Ломоносова*

Аннотация. DVM-система предназначена для разработки параллельных программ научно-технических расчетов на языках C-DVMH и Fortran-DVMH. Эти языки используют единую модель параллельного программирования (DVMH-модель) и являются расширением стандартных языков Си и Фортран спецификациями параллелизма, оформленными в виде директив компилятору. DVMH-модель позволяет создавать эффективные параллельные программы для гетерогенных вычислительных кластеров, в узлах которых в качестве вычислительных устройств наряду с универсальными многоядерными процессорами могут использоваться ускорители (графические процессоры или сопроцессоры Intel Xeon Phi). В статье описывается опыт успешного применения DVM-системы для разработки параллельного программного кода для расчетов задачи радиационной магнитной газодинамики и исследования динамики плазмы в канале КСПУ.

Ключевые слова: автоматизация разработки параллельных программ, DVM-система, Фортран, Си, плазменный ускоритель, радиационная магнитная газодинамика

Development of parallel software code for calculating the problem of radiation magnetic gas dynamics and the study of plasma dynamics in the channel of plasma accelerator

V.A. Bakhtin^{1,2}, D.A. Zakharov¹, A.N. Kozlov^{1,2}, V.S. Konovalov¹

¹ *Keldysh Institute of Applied Mathematics*

² *Lomonosov Moscow State University*

Abstract. DVM-system is designed for the development of parallel programs of scientific and technical calculations in C-DVMH and Fortran-DVMH languages. These languages use a single parallel programming model (DVMH model) and are extensions of the standard C and Fortran languages with parallelism specifications, written in the form of directives to the compiler. The DVMH model makes it possible to create efficient parallel programs for heterogeneous computing clusters, in the nodes of which accelerators (graphic processors or Intel Xeon Phi coprocessors) can be used as computing devices along with universal multi-core processors. The article describes the experience of successful use of DVM-system for the development of parallel software code for calculating the problem of radiation magnetic gas dynamics and the study of plasma dynamics in the channel of plasma accelerator.

Keywords: automation of development of parallel programs, DVM-system, GPU, Fortran, plasma accelerator, radiation magnetic gas dynamics

Современный уровень экспериментальных и численных исследований позволяет одновременно определять локальные значения макроскопических параметров плазмы и характеристики излучения. Это открывает новые возможности для проведения комплексных исследований, валидации моделей и сближения результатов расчетов с возможностями экспериментальных исследований. Этим обусловлена актуальность разработки численных моделей переноса излучения, рассматриваемых в качестве инжекторов для термоядерных установок. Исследование течений ионизирующегося газа и плазмы соответственно для первой и второй ступеней квазистационарных плазменных ускорителей (далее КСПУ) проводятся на основе разработанных эволюционных моделей радиационной магнитной газодинамики, эффективное решение которых потребовало разработки параллельных программных кодов для расчетов на высокопроизводительных многопроцессорных вычислительных комплексах. Исследования потоков плазмы во второй ступени КСПУ на основе модели радиационной магнитной газодинамики (РМГД) позволили выявить условия, обеспечивающие генерацию потоков дейтерий-третиевой или D-T плазмы с термоядерными параметрами при разрядных токах до 1 МА во второй ступени КСПУ.

1. Математическая постановка задачи

Постановка задачи включает систему уравнений РМГД с учетом конечной проводимости среды, теплопроводности и излучения. При условии квазинейтральности $n_i = n_e = n$ и равенства $\mathbf{V}_i = \mathbf{V}_e = \mathbf{V}$ для полностью ионизованной плазмы имеем

$$\frac{\partial \rho}{\partial t} + \operatorname{div}(\rho \mathbf{V}) = 0, \quad \rho \frac{d \mathbf{V}}{d t} + \nabla P = \frac{1}{c} \mathbf{j} \times \mathbf{H}, \quad \frac{d}{d t} = \frac{\partial}{\partial t} + (\mathbf{V}, \nabla), \quad (1)$$

$$\frac{\partial}{\partial t}(\rho \varepsilon) + \operatorname{div}(\rho \varepsilon \mathbf{V}) + P \operatorname{div} \mathbf{V} = \frac{\mathbf{j}^2}{\sigma} - \operatorname{div} \mathbf{q} - \operatorname{div} \mathbf{W}, \quad \varepsilon = 2 c_V T, \quad \mathbf{q} = -\kappa \nabla T,$$

$$\frac{\partial \mathbf{H}}{\partial t} = \text{rot}(\mathbf{V} \times \mathbf{H}) - c \text{rot} \frac{\mathbf{j}}{\sigma}, \quad \mathbf{j} = \frac{c}{4\pi} \text{rot} \mathbf{H},$$

где $\rho = m_i n$ – плотность, $P = P_i + P_e = 2 k_B n T$ – суммарное давление, \mathbf{q} – тепловой поток, κ – теплопроводность, $\sigma = e^2 n_e / m_e \nu_e$ – электропроводность среды, \mathbf{W} – поток энергии излучения, определяемый с помощью соотношений (3) и (4).

В данном случае исследования динамики потоков плазмы во второй ступени КСПУ проведены при наличии основной азимутальной компоненты магнитного поля в канале. Ускорение плазмы обеспечивает сила Ампера $\frac{1}{c} \mathbf{j} \times \mathbf{H}$, где \mathbf{j} — ток в плазме, имеющий преимущественно радиальное направление.

В качестве единиц измерения взяты длина канала L , характерная концентрация или плотность газа на входе в канал ускорителя n_0 ($\rho_0 = m n_0$) и температура T_0 . Характерная величина азимутального магнитного поля на входе в канал H_0 определяется разрядным током в системе J_p так, что $H_0 = 2 J_p / c R_0$, где R_0 – характерный радиус канала. С помощью этих величин формируются единицы: давления $P_0 = H_0^2 / 4\pi$, скорости $V_0 = H_0 / \sqrt{4\pi \rho_0}$, времени $t_0 = L / V_0$, тока в плазме $j_0 = c H_0 / 4\pi L$. В модели участвуют безразмерные параметры $\beta = 8\pi P_0 / H_0^2$ и $\text{Re}_m = \nu^{-1} = \sigma_0 T^{3/2}$, а также безразмерные значения теплопроводности и потока \mathbf{W} .

Постановка МГД задачи включает традиционные граничные условия. На входе в канал при $z = 0$ плазма подается с известными значениями плотности $\rho(r) = f_1(r)$ и температуры $T(r) = f_2(r)$. Считаем, что ток не меняется и поступает в систему только через электроды, т.е. $j_z = 0$ при $z = 0$ или $r H_\varphi = r_0 = \text{const}$, где $r_0 = R_0 / L$. Граничные условия на электродах $r = r_a(z)$ и $r = r_k(z)$, образующих стенки канала, предполагают их эквипотенциальность ($E_\tau = 0$) и непроницаемость поверхности для плазмы ($V_n = 0$). На оси системы при $r = 0$ имеем: $V_r = 0$, $V_\varphi = 0$, $H_\varphi = 0$.

Алгоритм численного решения уравнений (1) предполагает отображение расчетной области на единичный квадрат в плоскости (y, z) с помощью соотношения

$$r = (1 - y) r_k(z) + y r_a(z). \quad (2)$$

Для расчета гиперболической части МГД-уравнений используется разностная схема с коррекцией потоков. Магнитная вязкость и теплопроводность учитываются с помощью метода потоковой прогонки. Квазистационарные течения рассчитываются методом установления. Задача о переносе излучения решена в рамках разработанной 3D модели.

Поскольку скорость распространения излучения существенно выше скоростей плазмодинамических процессов, интенсивность излучения вычисляется с помощью стационарного уравнения переноса излучения,

$$\boldsymbol{\Omega} \cdot \nabla I_\nu(\mathbf{r}, \boldsymbol{\Omega}) = \eta_\nu(\mathbf{r}) - \kappa_\nu(\mathbf{r}) \cdot I_\nu(\mathbf{r}, \boldsymbol{\Omega}), \quad (3)$$

где $I_\nu(\mathbf{r}, \boldsymbol{\Omega})$ – интенсивность излучения с частотой ν , распространяющегося в направлении телесного угла $\boldsymbol{\Omega}$, отвечает точке с координатой \mathbf{r} . В свою очередь плотность энергии излучения U и плотность потока энергии излучения \mathbf{W} определяются через интенсивность излучения с помощью следующих соотношений

$$U(\mathbf{r}) = \frac{1}{c} \int_0^\infty \int_0^{4\pi} I_\nu(\mathbf{r}, \boldsymbol{\Omega}) d\boldsymbol{\Omega} d\nu, \quad \mathbf{W}(\mathbf{r}) = \int_0^\infty \int_0^{4\pi} I_\nu(\mathbf{r}, \boldsymbol{\Omega}) \boldsymbol{\Omega} d\boldsymbol{\Omega} d\nu. \quad (4)$$

Коэффициент поглощения $\kappa_\nu(\mathbf{r})$ и излучательная способность $\eta_\nu(\mathbf{r})$ являются известными функциями температуры, плотности вещества и представляют собой суммы из трех частей, отвечающих а) поглощению и излучению в линиях; б) фотоионизации и фоторекомбинации и в) тормозному излучению и обратному тормозному поглощению.

В силу соотношений (4) задачу о переносе излучения необходимо решать в трехмерной постановке задачи.

В данном случае сетка для 3D задачи о переносе излучения генерируется поворотом исходной сетки в плоскости переменных (z, r) на 360 градусов вокруг оси канала с заданным шагом. Для каждого узла или ячейки трехмерной сетки также строится дополнительная угловая сетка по азимутальному и полярному углу. Разбиение телесного угла на элементы угловой сетки производится методом, обеспечивающим равномерное распределение лучей по направлениям. Использовано 440 лучей в полном телесном угле $\boldsymbol{\Omega} = 4\pi$. В соответствии с методом длинных характеристик для решения уравнения переноса излучения (3) осуществляется трассировка лучей с целью определить точки их пересечения с гранями ячеек трехмерной координатной сетки и место падения лучей на стенки канала ускорителя и границы трехмерной расчетной области. Невидимые теневые области исключаются из расчета потока энергии излучения. Коэффициент поглощения $\kappa_\nu(\mathbf{r})$ и излучательная способность $\eta_\nu(\mathbf{r})$ вычисляются по среднему значению плотности и температуры в центре ячейки. Интенсивность вдоль лучей или характеристик, проходящих через любое количество однородных областей с известными коэффициентами κ_ν и η_ν , определяется в результате сшивки решений для однородных областей.

Детальная постановка задачи и используемые методы численного решения уравнения переноса излучения изложены в работах [1-3].

2. Разработка параллельной версии программы

Параллельный программный код для исследования высокоскоростных потоков плазмы в каналах квазистационарных плазменных ускорителей реализован с помощью DVM-системы [4,5] на языке Fortran-DVMH.

Остановимся на особенностях распараллеливания данного программного комплекса. Основная сложность разработки параллельной программы для кластера - необходимость принятия глобальных решений по распределению данных и вычислений с учетом свойств всей программы, а затем выполнения кропотливой работы по модификации программы и ее отладке. Большой объем программного кода, многомодульность, многофункциональность затрудняет принятие решений по согласованному распределению данных и вычислений.

Для облегчения процесса разработки параллельной версии программы можно использовать инкрементальное распараллеливание. Идея этого метода заключается в том, что распараллеливанию подвергается не вся программа целиком, а ее части (области распараллеливания) - в них заводятся дополнительные экземпляры требуемых данных, производится распределение этих данных и соответствующих вычислений. Области выбираются на основе времен, полученных с помощью профилирования последовательной программы.

Для взаимодействия с теми частями программы, которые не подвергались распараллеливанию, используются операции копирования исходных (нераспределенных) данных в дополнительные (распределенные) данные и обратно. Таким образом можно изолировать интересующие нас участки кода и распараллеливать каждую область отдельно от всей остальной программы. Это позволяет тестировать различные подходы к распределению данных и вычислений внутри области, которые могут существенно менять структуру хранения данных, но при этом любые изменения внутри области не будут требовать никаких модификаций кода вне области. Вместе с этим найти наиболее оптимальное распределение данных и вычислений в небольшой локальной области значительно проще, чем для всей программы.

После распараллеливания отдельных областей выполняется оптимизация параллельной программы, направленная на минимизацию объема копируемых данных. Для этого несколько областей могут объединяться в одну, если для общей области можно найти эффективное общее распределение. Также возможно добавление в область менее времяемких фрагментов, если это уменьшит количество операций копирования данных. Это позволяет минимизировать объем кода программы для анализа и распараллеливания – достаточно рассматривать только те фрагменты, которые работают с данными, уже использующимися внутри распараллеленной области.

Наличие оптимальных решений по распределению данных и вычислений в локальных областях немного упрощает нахождение общего распределения для объединенной области. Однако в случае, когда для объединенной области не удастся найти оптимальное общее распределение, инкрементальное

распараллеливание позволяет сделать процесс нахождения локальных оптимальных распределений значительно проще. Это очень полезно в случае, если программа состоит из нескольких частей, данные в которых существенно отличаются по структуре.

Для решения поставленной задачи используется сразу несколько различных математических моделей, поэтому было решено использовать именно инкрементальное распараллеливание, которое было успешно применено. Наиболее времяемкие части программы были определены при помощи анализатора производительности, который является частью DVM-системы. С точки зрения времени выполнения команд самой существенной частью программного комплекса была функция `run_radiat`, отвечающая расчету переноса излучения в 3D постановке задачи. Кроме того, значительное время требовалось на выполнение итерационного цикла в той части программного комплекса, который отвечает расчетам осесимметричных течений на основе МГД модели.

Опишем процесс распараллеливания основного итерационного цикла и функции `run_radiat`.

Итерационный цикл представлял собой обычный цикл с фиксированным количеством итераций. Внутренние циклы были описаны при помощи меток (рис. 1):

```
do 43 L = 1,NZ
  do 43 M = 1,NR
    HFI(M,L) = HRFI(M,L) / RAD(M,L)
  continue
43
```

Рис. 1. Фрагмент исходной программы

Все внутренние циклы для удобства были переписаны без использования меток, а также была произведена замена всех используемых в них массивов (рис. 2). Отказ от использования меток был чисто техническим преобразованием с целью повышения уровня читаемости программы и облегчения распараллеливания.

```
do L = 1,NZ
  do M = 1,NR
    new_HFI(M,L) = new_HRFI(M,L) / new_RAD(M,L)
  end do
end do
```

Рис. 2. Фрагмент программы после преобразования

Замена массивов производилась как подготовка к дальнейшему распределению данных массивов в изоляции от остальной части программы. Так как итерационный цикл - лишь некоторая часть всего комплекса, было необходимо удостовериться, что распределение массивов не потребует вносить никаких изменений в остальную часть программы. Всего таким образом было заменено около 45 массивов. Все эти массивы, как оригинальные, так и копии, были заменены на динамические, и их выделение происходило перед самым

началом итерационного цикла. Оригинальные массивы в этот момент наоборот уничтожались, чтобы не потреблять лишнюю память (рис. 3):

```

allocate(new_HRFI)
new_HRFI = HRFI
deallocate(HRFI)
<итерационный цикл>
allocate(HRFI)
HRFI = new_HRFI
deallocate(new_HRFI)

```

Рис. 3. Работа с копиями массивов

Указанные выше переходы от одного массива к другому ставились перед итерационным циклом и после него в случае, если данные, имевшиеся в этих массивах использовались соответственно внутри итерационного цикла или после него.

Большая часть циклов имела достаточно простой характер обращений - либо к соответствующим для витка по индексам элементам, либо к ближайшим соседям (рис. 4):

```

do L = 2,NZM1
do M = 2,NRM1
FGB = (GAM - 1.) / BB * new_RDY(L) / new_TEM(M,L)
DRHDY = (new_HRFI(M + 1,L) - new_HRFI(M - 1,L)) / DY2
DRHDZ = (new_HRFI(M,L + 1) - new_HRFI(M,L - 1)) / DZ2
! more code
end do
end do

```

Рис. 4. Типовой цикл

Для таких массивов, как new_HRFI из примера выше, при распределении были указаны соответствующие теневые грани (спецификация SHADOW).

подавляющее большинство циклов не было тесно-вложенными, что не позволило провести их распределение сразу по двум измерениям. Провести распределение по какому-либо одному распределению также было нельзя, так как присутствовали циклы с прямыми зависимостями как по одному, так и по другому измерению (рис. 5):

```

do L = 2,NZM1
! more code
do N = 2,NRM1
M = NR - N + 1
WL(M) = ((1. - C(M) / A(M) * BET(M)) * (B(M) * WL(M+1) - F(M)) + C(M) * GAMM(M)) / A(M)
PROR(M) = (A(M) * GAMM(M) + BET(M) * (F(M) - B(M) * WL(M + 1))) / A(M)
end do
!more code
end do

```

Рис. 5. Цикл с зависимостями

Аналогичные циклы были и по другому измерению. Значительное число циклов также имело существенный объем обращений к потенциально удаленным данным по одному из измерений (рис. 6):

```
do L = 2,NZM1
  ! more code
  PPL = VR1 * new_RAD(1,L) * new_RDY(L) * new_PLT(1,L)
  !more code
end do
```

Рис. 6. Потенциально удаленные данные

Для решения этих проблем было решено использовать сразу два шаблона распределения данных (спецификация TEMPLATE) - по одному на каждое измерение. В одном случае получалось, что каждый процесс получал часть «строк» двумерного массива и работал с ними. В другом - каждый процесс получал часть «столбцов» двумерного массива и обрабатывал их. Для каждого цикла выбиралось измерение без зависимостей и без обращений к удаленным данным, и распараллеливание производилось по нему. Созданные ранее копии массивов также распределялись по тем измерениям, по которым распараллеливался цикл, в котором они использовались. Если массив использовался в циклах, среди которых были параллельные циклы как по одному, так и по другому измерению, для него создавалось 2 копии сразу. Между циклами, распараллеленными по разным измерениям, при необходимости производилось переключение между шаблонами (рис. 7):

```
!DVM$ PARALLEL(L) ON new_PLT(*,L), PRIVATE(M)
do L = 2,NZ
  do M = 1,NR
    ! more code
  end do
end do
new2_PLT = new_PLT
!DVM$ PARALLEL (M) ON new2_PLT(M,*), PRIVATE(L)
do M = 2,NR
  do L = 1,NZ
    ! more code
  end do
end do
```

Рис. 7. Переключение между шаблонами

Серьезных переключений в программе оказалось лишь 2 - перевод 8 массивов с распределения по второму измерению на распределение по первому, и перевод этих же 8 массивов назад. Еще один раз в программе потребовался аналогичный перевод туда и обратно, но уже лишь для одного массива. По созданным шаблонам было распределено около 30 массивов, из которых около 20 имело копии для обоих шаблонов распределения. Также по ним было распараллелено более 50 разных циклов.

В функции `run_radiat` было проведено распараллеливание 5 циклов. Эти циклы использовали разнообразные структуры данных и массивы уникальных форматов (рис. 8):

```

do inode = 1,mesh3d%NNodes
  den3d%fval(inode) = 0.e0_r8p
  tem3d%fval(inode) = 0.e0_r8p
  do i = 1,valsInterp%ielem(inode)%nSrc
    n1 = valsInterp%ielem(inode)%iSrc(i)
    eps = valsInterp%ielem(inode)%wSrc(i)
    den3d%fval(inode) = den3d%fval(inode) + den2d%fval(n1)*eps
    tem3d%fval(inode) = tem3d%fval(inode) + tem2d%fval(n1)*eps
  end do
end do

```

Рис. 8. Использование сложных структур данных

Так как основные массивы в этих циклах имели уникальную структуру, которые нельзя было хорошо соотнести друг с другом, для всех 5 циклов было создано 4 разных шаблона распределения (т.к. 2 из 5 циклов удалось распараллелить, используя один и тот же шаблон). Все распределяемые массивы были заменены на обычные массивы стандартных типов, остальные массивы и структуры данных были оставлены в изначальном виде. Итоговый вариант цикла из примера выше после преобразований и распараллеливания выглядел так (рис. 9):

```

!DVM$ PARALLEL(inode) ON new_den3d(inode),PRIVATE(n1,eps,i)
do inode = 1,m3d
  new_den3d(inode) = 0.e0_r8pч
  new_tem3d(inode) = 0.e0_r8p
  do i = 1,valsInterp%ielem(inode)%nSrc
    n1 = valsInterp%ielem(inode)%iSrc(i)
    eps = valsInterp%ielem(inode)%wSrc(i)
    new_den3d(inode) = new_den3d(inode) + new2_den2d(n1)*eps
    new_tem3d(inode) = new_tem3d(inode) + new2_tem2d(n1)*eps
  end do
end do

```

Рис. 9. Цикл после преобразований

Главной сложностью распараллеливания этой части программы была косвенная адресация, выделенная в примерах выше. Ее наличие уже означает, что любой виток цикла может обращаться к любому элементу адресуемого косвенно массива. Ее форма в данной задаче такова, что каждый виток может обращаться к произвольному числу разных элементов адресуемого массива, возможно даже ко всем. Более того, ситуация, когда каждый процесс будет запрашивать практически все элементы косвенно адресуемого массива (то есть значения `n1` для каждого набора витков, выделяемых процессу, проходят через практически все возможные элементы массивов `den2d%fval` и `tem2d%fval`) для данной задачи является нормой. Поэтому было решено проводить запись данных в распределенные массивы, а если в последующих циклах этот массив

адресуется косвенно - производилось его размножение по всем процессам. Учитывая то, что каждый процесс использует почти каждый элемент массива - лишние накладные расходы на совершение полного копирования являются незначительными. После цикла, указанного выше, например, происходило размножение в виде (рис. 10):

```
new2_den3d = new_den3d
new2_tem3d = new_tem3d
```

Рис. 10. Копирование данных после выполнения цикла

Префикс `new_` указывал на распределенные массивы, а `new2_` - на размноженные. Отдельной проблемой было наличие редукции по сумме с косвенной адресацией (рис. 11):

```
do idir = 1,NDirs
  do iEn = 1,NEnGroups
    do isegm = 1,NPoints
      inode = pTrace%rayTree(idir)%segm(isegm)%inode
      !more code
      cUden%fval(inode) = cUden%fval(inode) + cU_ptX
    end do
  end do
end do
```

Рис. 11. Цикл с редукцией

В данном случае получалось, что любой виток цикла мог производить запись, которая еще и суммировалась, в любой элемент массива. Для решения этого вопроса был создан специальный распределенный массив, позволявший каждому витку проводить суммирование в своем собственном «столбце». После этого уже производилось суммирование по последнему измерению (рис. 12):

```
do idir = 1,NDirs
  !DVM$ PARALLEL(iEn) ON new_cuden3d(*,iEn)
  do iEn = 1,NEnGroups
    do isegm = 1,NPoints
      inode = pTrace%rayTree(idir)%segm(isegm)%inode
      !more code
      new_cuden3d(inode,iEn) = new_cuden3d(inode,iEn) + cU_ptX
    end do
  end do
end do
!DVM$ PARALLEL(iEn) ON uni_cuden3d(*,iEn), PRIVATE(inode),
!DVM$* REDUCTION(SUM(new2_cuden3d))
do iEn = 1,NEnGroups
  do inode = 1,m3d
    new2_cuden3d(inode)=new2_cuden3d(inode)+ new_cuden3d(inode,iEn)
  end do
end do
```

Рис. 12. Реализации редукции

В итоге каждый процесс получал корректную копию редуцированного массива.

По итогам распараллеливания была получена версия программы, которую возможно выполнять на кластерах, в узлах которых установлены многоядерные процессоры.

3. Эффективность параллельной версии программы

Расчеты с помощью параллельной версии программы проводились на двух многопроцессорных высокопроизводительных вычислительных комплексах К-100 (ИПМ РАН) и MVS1P5 (МСЦ РАН). На рис. 13 и 14 представлены графики изменения времени выполнения программы в секундах в зависимости от числа используемых ядер данных вычислительных комплексов. Речь идет о расчетах на один временной шаг, определяемый условием Куранта в решении эволюционной задачи на основе РМГД модели. Кривые 1 на рисунках отвечают расчетам с помощью метода длинных характеристик, который требует наибольших затрат вычислительных ресурсов в случае решения полностью трехмерной задачи и расчету интегральных характеристик во всех узлах трехмерной координатной сетки. Кривые 2 и 3 соответствуют расчетам интегральных характеристик излучения в одной плоскости переменных (r, z) с учетом аксиальной симметрии течения. Наряду с методом длинных характеристик был реализован метод коротких характеристик, которому отвечают кривые 2. Метод коротких характеристик приводит к численной диффузии при расчете поля излучения, и он требует больше времени для расчета по сравнению с методом длинных характеристик, который представлен кривыми 3 на рисунках при условии вычисления интегральных характеристик излучения в одной плоскости переменных (r, z) с учетом аксиальной симметрии течения.

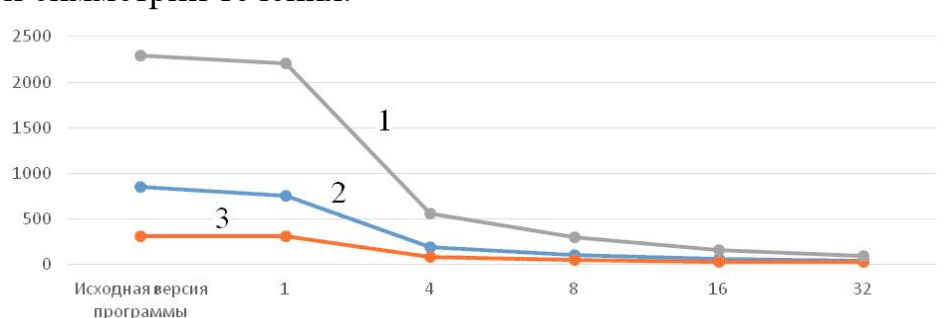


Рис. 13. Время выполнения программы в секундах на различном числе ядер К-100

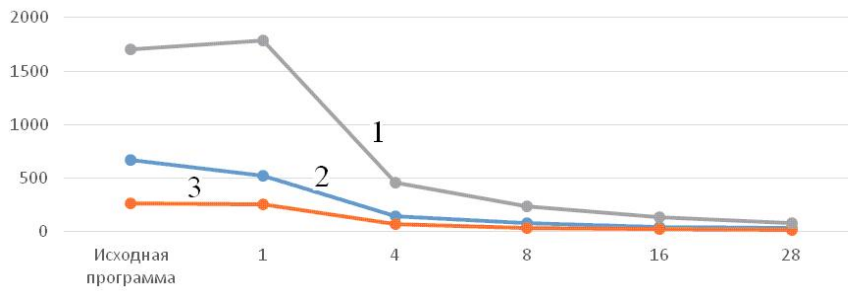


Рис. 14. Время выполнения программы в секундах на различном числе ядер MVS1P5

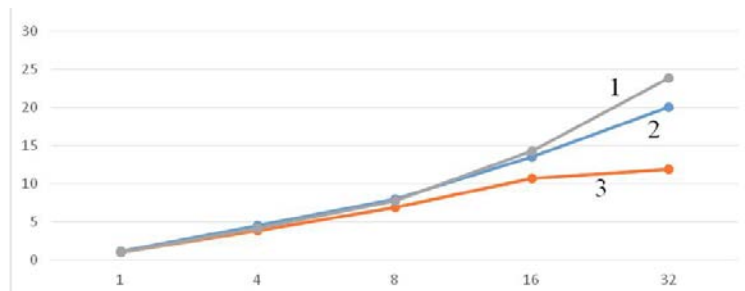


Рис. 15. Ускорение процесса выполнения программы относительно исходной версии на различном числе ядер K-100

Рис. 15 и 16 показывают, как меняется ускорение процесса выполнения параллельной версии программы относительно исходной последовательной версии на различном числе ядер вычислительных комплексов соответственно в ИПМ РАН и МСЦ РАН. Кривые 1, 2 и 3 на данных рисунках отвечают тем же условиям расчетов, указанным выше.

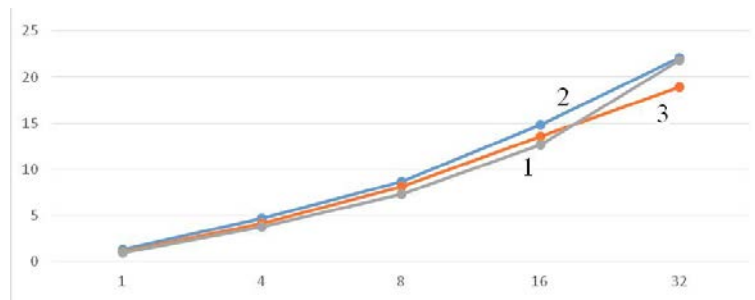


Рис. 16. Ускорение процесса выполнения программы относительно исходной версии на различном числе ядер MVS1P5

Более слабое ускорение на K-100 для кривой 3 на рис. 15 для 32 процессоров объясняется тем, что время работы программы уже очень мало, менее 30 секунд, и накладные расходы начинают занимать существенный процент времени работы программы.

Ускорение параллельной версии программы на одном процессоре можно объяснить за счет более оптимального характера обращений к памяти. Замена части структур, использующих указатели, на набор обычных массивов привела к уменьшению общего количества обращений к памяти. Это также позволило в

некоторых случаях избавиться от косвенной адресации, что увеличило эффективность использования кэш-памяти. Получаемое ускорение существенно зависит от используемой сетки, режима работы программы и характеристик памяти конкретной машины. Его может быть достаточно для того, чтобы покрыть все возникающие при распараллеливании накладные расходы и даже немного ускорить программу на одном процессоре.

Заключение

С помощью DVM-системы разработан параллельный программный код для расчетов задачи радиационной магнитной газодинамики и исследования динамики плазмы в канале КСПУ. При этом были достигнуты следующие результаты по ускорению процесса выполнения программы относительно исходной версии: до 24 раз на 32 процессорах вычислительного комплекса К-100 (ИПМ РАН) и до 22 раз на 28 процессорах вычислительной системы MVS1P5 (МСЦ РАН).

Многочисленные расчеты, выполненные с помощью параллельной версии программы, показали, что значение разрядного тока, необходимое для достижения энергии ионов $\varepsilon_i = 30 \text{ KeV}$, увеличивается пропорционально размерам установки. Было установлено, что уменьшение характерной концентрации плазмы n_0 на входе в канал ускорителя позволяет существенно уменьшить значения разрядных токов. Были определены значения разрядного тока в установке, обеспечивающие на выходе энергию ионов на уровне 30 KeV , который необходим для последующей реакции синтеза D-T плазмы в магнитных ловушках для удержания плазмы. Найденные разрядные токи являются вполне приемлемыми для существующих установок КСПУ и не приводят к оплавлению элементов конструкций.

Исследование выполнено за счет гранта Российского научного фонда (проект № 16-11-10278).

Литература

1. Kozlov A. N., Konovalov V. S. Numerical study of the ionization process and radiation transport in the channel of plasma accelerator // Communications in Nonlinear Science and Numerical Simulation, 2017, Vol. 51 — P. 169-179 .
2. Kozlov A. N. The study of plasma flows in accelerators with thermonuclear parameters // Plasma Physics and Controlled Fusion, 2017, Vol. 51, No. 11, Ar. 115004 — P. 1-7. — 10.1088/1361-6587/aa86be .
3. Kozlov A. N., Konovalov V. S. Radiation transport in the ionizing gas flow in the quasi-steady plasma accelerator // Journal of Physics: Conference Series, 2018, Vol. 946 — 10.1088/1742-6596/946/1/012165 .

4. Язык C-DVMH. C-DVMH компилятор. Компиляция, выполнение и отладка CDVMH-программ. — URL: http://dvm-system.org/static_data/docs/CDVMH-reference-ru.pdf .
5. Язык Fortran-DVMH. Fortran-DVMH компилятор. Компиляция, выполнение и отладка DVMH-программ. — URL: http://dvm-system.org/static_data/docs/FDVMH-user-guide-ru.pdf .

References

1. Kozlov A. N., Konovalov V. S. Numerical study of the ionization process and radiation transport in the channel of plasma accelerator // Communications in Nonlinear Science and Numerical Simulation, 2017, Vol. 51 — P. 169-179 .
2. Kozlov A. N. The study of plasma flows in accelerators with thermonuclear parameters // Plasma Physics and Controlled Fusion, 2017, Vol. 51, No. 11, Ar. 115004 — P. 1-7. — 10.1088/1361-6587/aa86be .
3. Kozlov A. N., Konovalov V. S. Radiation transport in the ionizing gas flow in the quasi-steady plasma accelerator // Journal of Physics: Conference Series, 2018, Vol. 946 — 10.1088/1742-6596/946/1/012165 .
4. C-DVMH language, C-DVMH compiler, compilation, execution and debugging of DVMH programs. — URL: http://dvm-system.org/static_data/docs/CDVMH-reference-en.pdf .
5. Fortran DVMH language, Fortran DVMH compiler, compilation, execution and debugging of DVMH programs. — URL: http://dvm-system.org/static_data/docs/FDVMH-user-guide-en.pdf .