Опыт решения прикладных задач, использующих нерегулярные сетки, с использованием DVM-системы

В.А. Бахтин, Д.А. Захаров, В.А. Крюков,
Н.В. Поддерюгина, М.Н. Притула

dvm@keldysh.ru

Ростов-на-Дону, 5 апреля, 2018

Работа поддержана грантами РФФИ № 16-07-01014, 16-07-01067 и 17-01-00820

Федеральное государственное учреждение
ФЕДЕРАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЦЕНТР
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ ИМ. М.В. КЕЛДЫША
Российской академии наук

План доклада

- Автоматизация разработки параллельных программ
  - Система DVM
  - Система САПФОР
- Инкрементальное распараллеливание с использованием DVM-системы
- Распараллеливание прикладных задач, использующих неструктурированные сетки
  - Программа HyperbolicSolver2D
  - Задача теплопроводности в шестиграннике
  - Программа для решения задачи газовой динамики методом Галеркина
  - Программа для расчета течений плазмы с учетом переноса излучения на основе РМГД модели

Состав DVM-системы

Система состоит из следующих компонент:

- Компилятор Fortran-DVMH
- Компилятор C-DVMH
- Библиотека поддержки LIB-DVMH
- DVMH-отладчик
- Анализатор производительности DVMH-программ

Средства программирования

C-DVMH = Язык Си(C++) + специальные прагмы

Fortran-DVMH = Язык Фортран 95 + специальные комментарии

- Специальные комментарии и прагмы являются высокоуровневыми спецификациями параллелизма в терминах последовательной программы
- Отсутствуют низкоуровневые передачи данных и синхронизации
- Последовательный стиль программирования
- Спецификации параллелизма «невидимы» для стандартных компиляторов
- Существует единственный экземпляр программы для последовательного и параллельного счета

Спецификации параллельного выполнения программы

- Распределение элементов массива между процессорами
- Распределение витков цикла между процессорами
- Спецификация параллельно выполняющихся секций программы (параллельных задач) и отображение их на процессоры
- Организация эффективного доступа к удаленным (расположенным на других процессорах/ускорителях) данным

Спецификации параллельного выполнения программы

- Организация эффективного выполнения редукционных операций - глобальных операций с расположенными на различных процессорах/ускорителях данными (таких, как их суммирование или нахождение их максимального или минимального значения)
- Определение фрагментов программы (регионов) для возможного выполнения на ускорителях
- Управление перемещением данных между памятью ЦПУ и памятью ускорителей
- Управление параллельным вводом-выводом

Инкрементальное распараллеливание с использованием DVM-системы

- Возможность распараллелить не всю программу, а ее времяемкие фрагменты.
- Позволяет найти лучшие схемы распараллеливания времяемких фрагментов.
- Постепенное добавление распараллеливаемых фрагментов с сохранением уже найденных решений.

Решение задачи для уравнения теплопроводности методом Якоби (512x512x512, NITER=10)

```
program jacobi
   double precision, allocatable, dimension(:,:,:) :: f, newf, r
   …
   allocate(f(mx, my, mz))
   allocate(newf(mx, my, mz))
   allocate(r(mx, my, mz))
   curf = 0
   do n = 1, NITER
      if (curf .eq. 0) then
         eps = dostep(f, newf, r, rdx2, rdy2, rdz2, beta, mx, my, mz)
      else
         eps = dostep(newf, f, r, rdx2, rdy2, rdz2, beta, mx, my, mz)
      endif
      print *, 'Iteration=' , n, 'eps=', eps
      curf = 1 - curf
    enddo
end
```

```
      double precision function dostep(f, newf, r, rdx2, rdy2, rdz2,
&        beta, mx, my, mz)
      integer :: mx, my, mz
      double precision, dimension(mx,my,mz) :: f, newf, r
      double precision :: rdx2, rdy2, rdz2, beta, eps
      integer :: i, j, k
      eps = 0.
      do k = 2, mz - 1
        do j = 2, my - 1
              do i = 2, mx - 1
                    newf(i, j, k) = ((f(i-1,j,k)+f(i+1,j,k))*rdx2
&                                   +(f(i,j-1,k)+f(i,j+1,k))*rdy2
&                             +(f(i,j,k-1)+f(i,j,k+1))*rdz2
&                                   -r(i,j,k)) * beta
                    eps = max(eps,abs(newf(i,j,k)-f(i,j,k)))
              enddo
          enddo
       enddo
       dostep = eps
       end function
```

Распараллеливание программы Якоби

- Перенос программы в другую ОС, на другую вычислительную систему. Отладка программы
- Определение времяемких фрагментов программы
- Распараллеливание времяемких фрагментов программы
- Добавление распараллеливаемых фрагментов с сохранением уже найденных решений

Сравнительная отладка программы

- Для поиска ошибок используется метод накопления и сравнения результатов вычислений, который позволяет определить место в программе и момент, когда появляются расхождения
- При трассировке вычислений выполняется сбор информации обо всех чтениях и модификациях переменных, о начале выполнения каждого витка цикла
- Степенью подробности и объемом трассировки можно управлять при конвертации программы опциями DVMH-конверторов, а также при выполнении программы
- Существует возможность управлять точностью при сравнении результатов
- Генерируются протоколы с различиями, обнаруженными в процессе сравнения и скачками в значениях переменных и элементах массивов

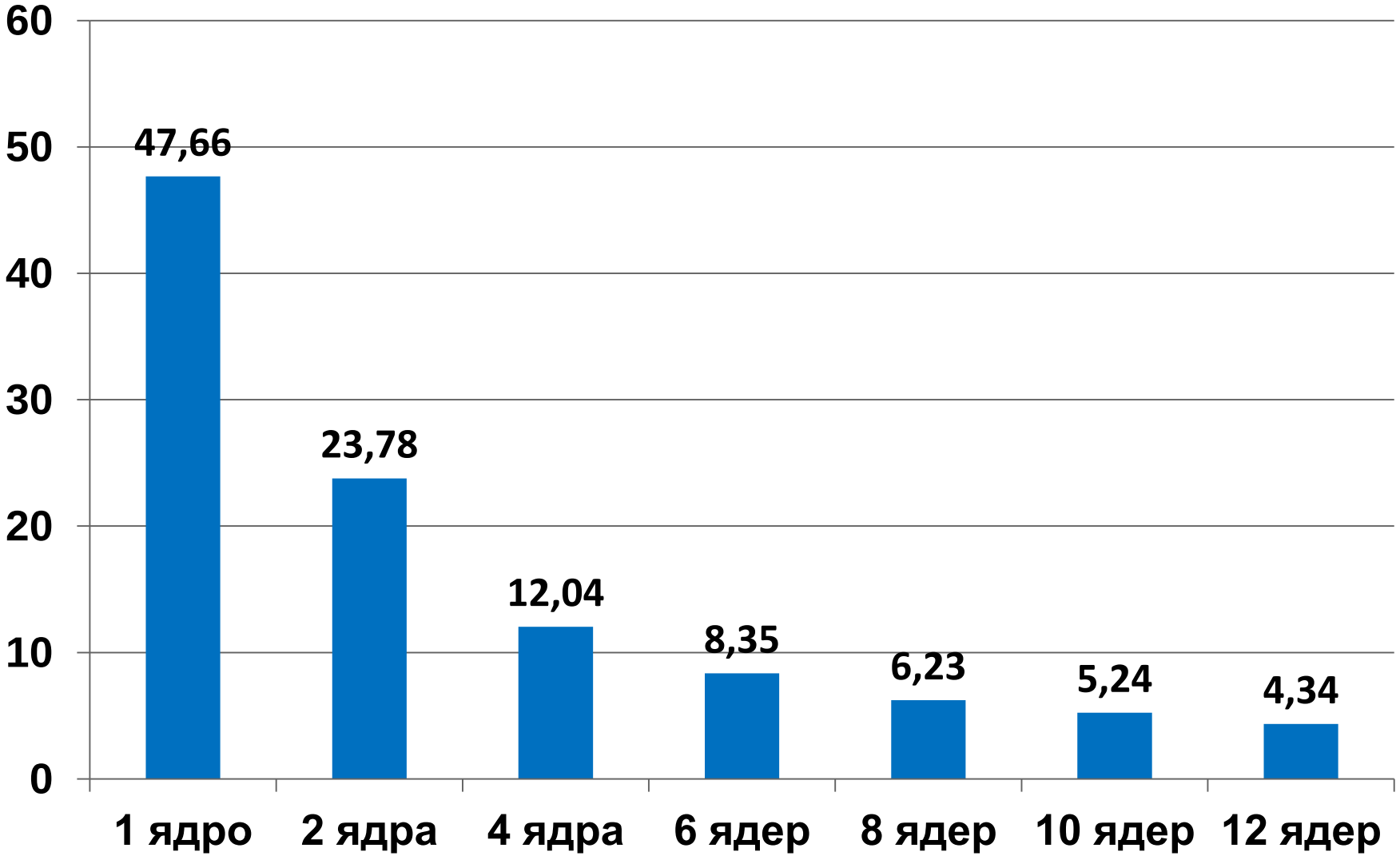
Определение времяемких фрагментов программы ./dvm f –e4 jacobi.f

```
INTERVAL ( NLINE=12 SOURCE=jacobi.f ) LEVEL=3 SEQ EXE_COUNT=5120
--- The main characteristics ---
Parallelization efficiency      1.0000
Execution time                  57.3396
Processors                      1
Threads amount                  1
Total time                      57.3396
Productive time                 57.3396 ( CPU= 56.8142 Sys= 0.5254 I/O= 0.0000 )
--- The comparative characteristics ---
                        Tmin N proc     Tmax N proc     Tmid
Execution time          57.3396    1    57.3396    1    57.3396
User CPU time           56.8142    1    56.8142    1    56.8142
Sys. CPU time           0.5254    1    0.5254    1    0.5254
Processors              1    1        1    1        1
--- The execution characteristics ---
                        1
Execution time          57.3396
User CPU time           56.8142
Sys. CPU time           0.5254
Processors              1
```

```
      double precision function dostep(f, newf, r, rdx2, rdy2, rdz2,
     &                 beta, mx, my, mz)
      integer :: mx, my, mz
      double precision, dimension(mx,my,mz) :: f, newf, r
      double precision :: rdx2, rdy2, rdz2, beta, eps
      integer :: i, j, k
      eps = 0.
CDVM$ PARALLEL (k,j,i), REDUCTION(max(eps))
      do k = 2, mz - 1
         do j = 2, my - 1
              do i = 2, mx - 1
                   newf(i, j, k) = ((f(i-1,j,k)+f(i+1,j,k))*rdx2
     &                             +(f(i,j-1,k)+f(i,j+1,k))*rdy2
     &                             +(f(i,j,k-1)+f(i,j,k+1))*rdz2
     &                             -r(i,j,k)) * beta
                   eps = max(eps,abs(newf(i,j,k)-f(i,j,k)))
              enddo
         enddo
      enddo
      dostep = eps
      end function
```

Программа для мультипроцессора

Времена выполнения программы в секундах на процессоре Intel Xeon E5-2660 (k10.kiam.ru)



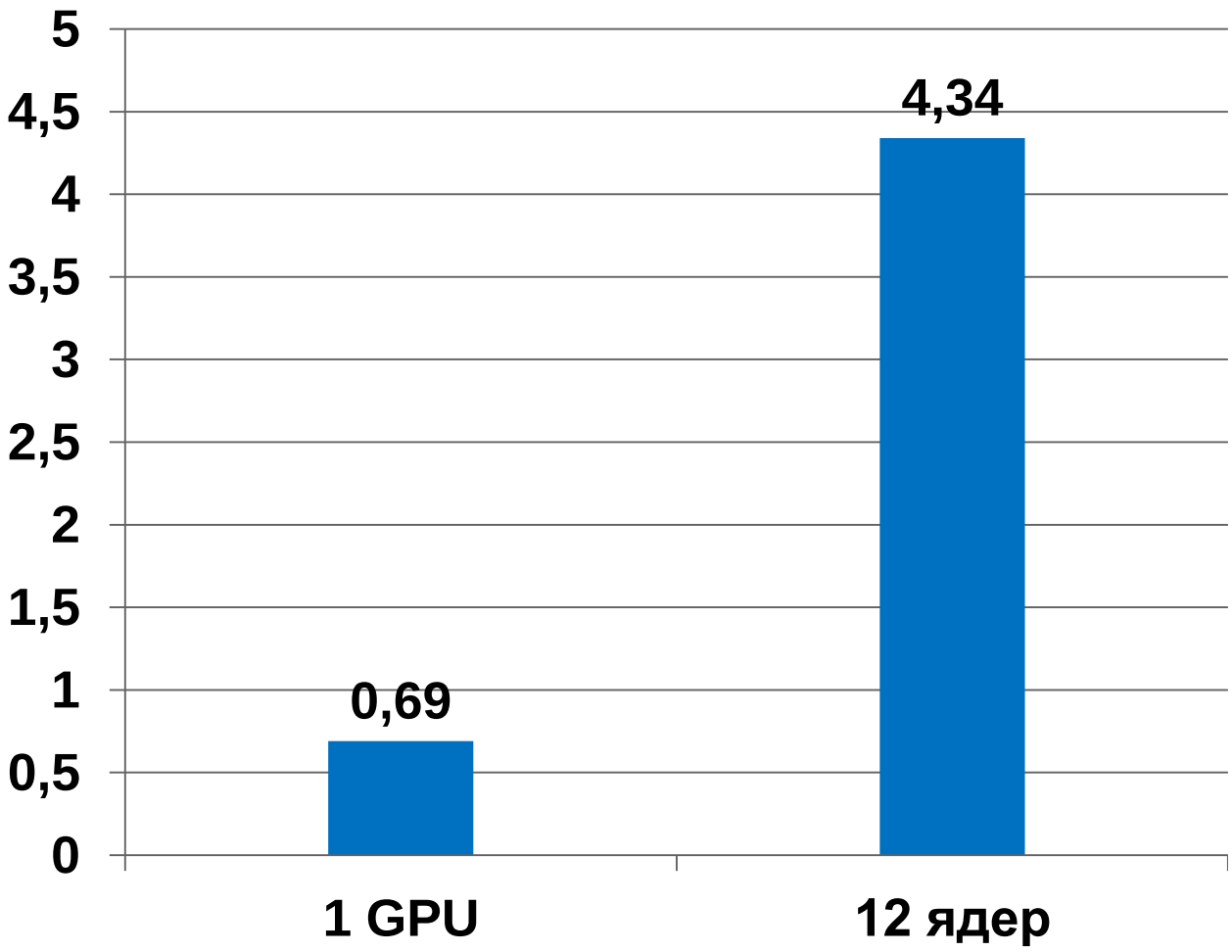
```
      double precision function dostep(f, newf, r, rdx2, rdy2, rdz2…)
      …
      eps = 0.
CDVM$ ACTUAL(eps)
CDVM$ REGION INOUT(f,newf, eps), IN(r,rdx2,rdy2,rdz2,beta)
CDVM$ PARALLEL (k,j,i), REDUCTION(max(eps))
      do k = 2, mz - 1
         do j = 2, my - 1
              do i = 2, mx - 1
                   newf(i, j, k) = ((f(i-1,j,k)+f(i+1,j,k))*rdx2
     &                            +(f(i,j-1,k)+f(i,j+1,k))*rdy2
     &                            +(f(i,j,k-1)+f(i,j,k+1))*rdz2
     &                            -r(i,j,k)) * beta
                   eps = max(eps,abs(newf(i,j,k)-f(i,j,k)))
              enddo
          enddo
       enddo
CDVM$ ENDREGION
CDVM$ GET_ACTUAL(eps)
       dostep = eps
       end function
```

Программа для графического ускорителя

```
      double precision function dostep(f, newf, r, rdx2, rdy2, rdz2…)
      …
      eps = 0.
CDVM$ ACTUAL(eps)
CDVM$ REGION
CDVM$ PARALLEL (k,j,i), REDUCTION(max(eps))
      do k = 2, mz - 1
         do j = 2, my - 1
              do i = 2, mx - 1
                   newf(i, j, k) = ((f(i-1,j,k)+f(i+1,j,k))*rdx2
     &                            +(f(i,j-1,k)+f(i,j+1,k))*rdy2
     &                            +(f(i,j,k-1)+f(i,j,k+1))*rdz2
     &                            -r(i,j,k)) * beta
                   eps = max(eps,abs(newf(i,j,k)-f(i,j,k)))
              enddo
          enddo
       enddo
CDVM$ ENDREGION
CDVM$ GET_ACTUAL(eps)
       dostep = eps
       end function
```

Программа для графического ускорителя

Времена выполнения программы в секундах на графическом процессоре nVidia Fermi M2090 и процессоре Intel Xeon E5-2660 (k10.kiam.ru)



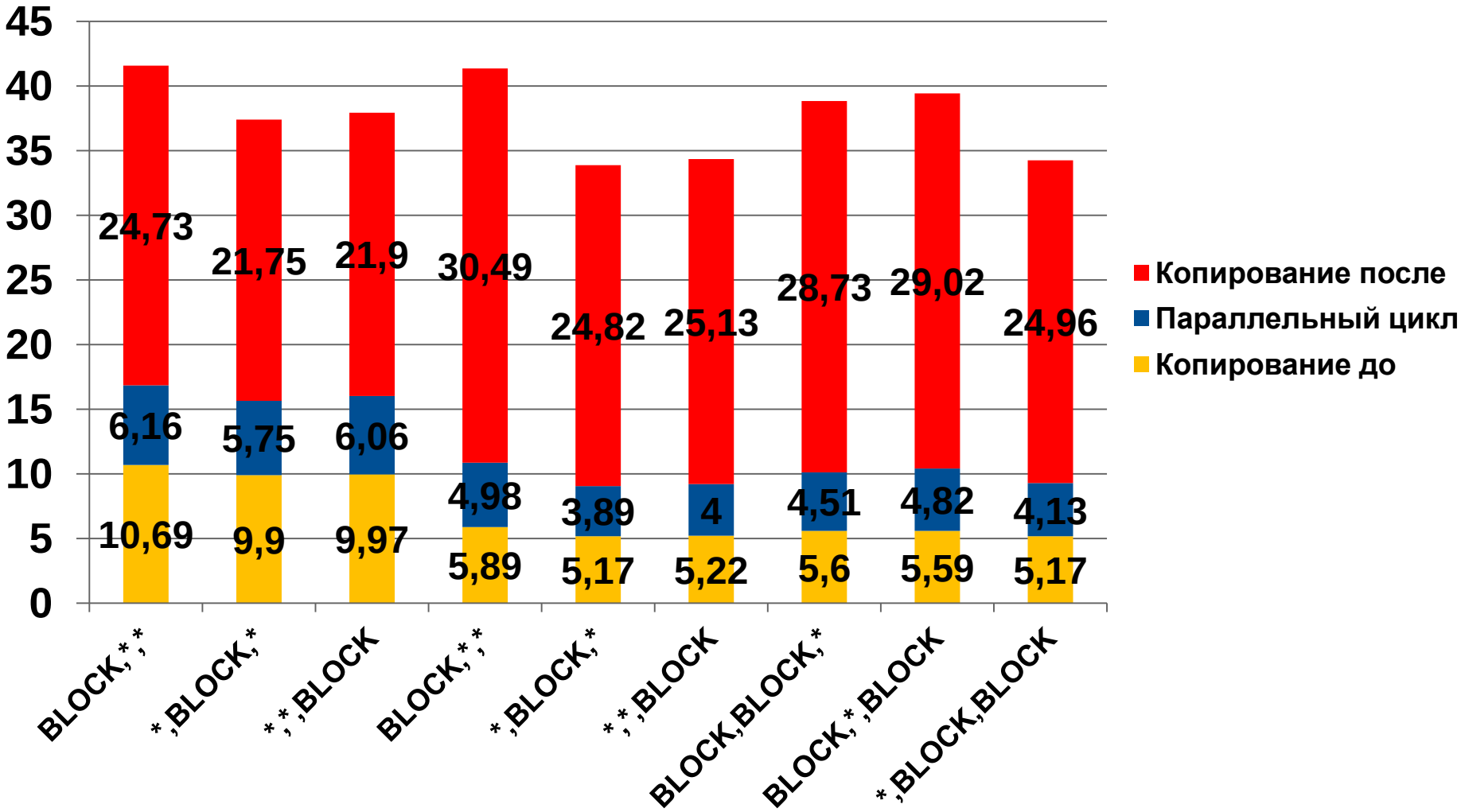
```
      double precision function dostep(f_, newf_, r_, rdx2, rdy2, rdz2..)
       ...
      double precision, dimension(mx,my,mz) :: f_, newf_, r_
C      МАССИВЫ КОПИИ
      double precision, dimension(mx,my,mz) :: f, newf, r
C      РАСПРЕДЕЛЕНИЕ ДАННЫХ
CDVM$ DISTRIBUTE (BLOCK, BLOCK,BLOCK) :: f
CDVM$ ALIGN newf(i,j,k) WITH f(i,j,k)
CDVM$ ALIGN r(i,j,k) WITH f(i,j,k)
CDVM$ INTERVAL(1)
C      КОПИРОВАНИЕ ДАННЫХ
       f = f_
       newf = newf_
       r = r_
CDVM$ END INTERVAL
      eps = 0.
```

Программа для кластера

```
CDVM$ PARALLEL (k,j,i) ON newf(i,j,k), REDUCTION(max(eps)),
CDVM$* SHADOW_RENEW(f)
          do k = 2, mz - 1
             do j = 2, my - 1
                 do i = 2, mx - 1
                      newf(i, j, k) = ((f(i-1,j,k)+f(i+1,j,k))*rdx2
   &                                  +(f(i,j-1,k)+f(i,j+1,k))*rdy2
   &                                  +(f(i,j,k-1)+f(i,j,k+1))*rdz2
   &                                  -r(i,j,k)) * beta
                      eps = max(eps,abs(newf(i,j,k)-f(i,j,k)))
                 enddo
              enddo
          enddo
CDVM$ INTERVAL(3)
C         КОПИРОВАНИЕ ДАННЫХ
          f_ = f
          newf_ = newf
          r_ = r
CDVM$ END INTERVAL
          dostep = eps
          end function
```

Программа для кластера

Времена выполнения частично распараллеленной программы в секундах на узле k10.kiam.ru для разных распределений массивов



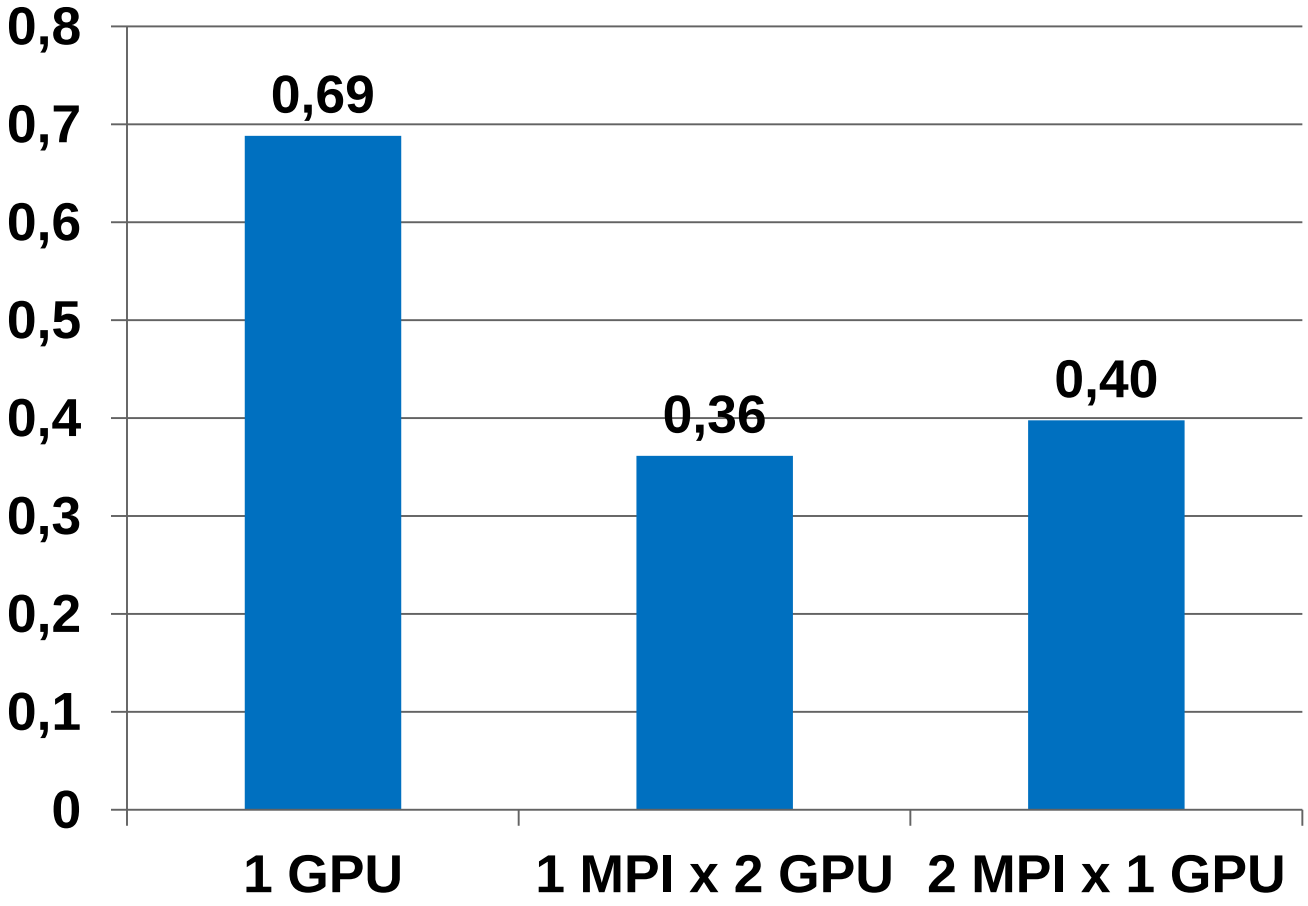
Итоговый вариант программы для кластера с ускорителями

```
      program jacobi
        double precision, allocatable, dimension(:,:,:) :: f, newf, r
CDVM$ DISTRIBUTE (BLOCK, BLOCK,BLOCK) :: f
CDVM$ ALIGN newf(i,j,k) WITH f(i,j,k)
CDVM$ ALIGN r(i,j,k) WITH f(i,j,k)
        …
        do n = 1, NITER
          if (curf .eq. 0) then
            eps = dostep(f, newf, r, rdx2, rdy2, rdz2, beta, mx, my, mz)
          else
            eps = dostep(newf, f, r, rdx2, rdy2, rdz2, beta, mx, my, mz)
          endif
          print *, 'Iteration=' , n, 'eps=', eps
          curf = 1 - curf
        enddo
      end
```

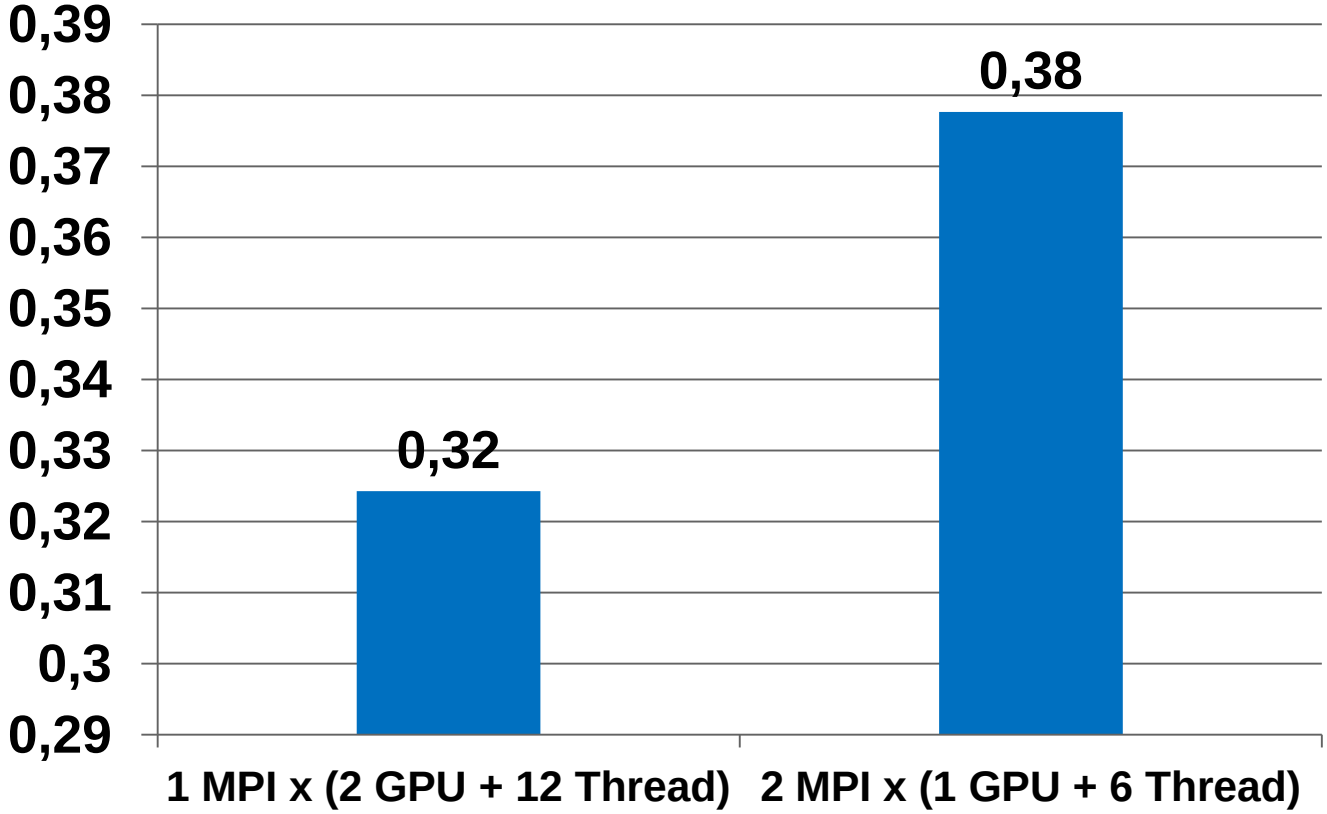
```
      double precision function dostep(f, newf, r, rdx2, rdy2, rdz2…)
CDVM$ INHERIT f,newf,r
      ...
      eps=0.
CDVM$ ACTUAL(eps)
CDVM$ REGION
CDVM$ PARALLEL(k,j,i) ON newf(i,j,k),REDUCTION(max(eps)),
CDVM$*  SHADOW_RENEW(f)
      do k = 2, mz - 1
         do j = 2, my - 1
              do i = 2, mx - 1
                   newf(i, j, k) = ((f(i-1,j,k)+f(i+1,j,k))*rdx2
     &                            +(f(i,j-1,k)+f(i,j+1,k))*rdy2
     &                            +(f(i,j,k-1)+f(i,j,k+1))*rdz2
     &                            -r(i,j,k)) * beta
                   eps = max(eps,abs(newf(i,j,k)-f(i,j,k)))
              enddo
         enddo
      enddo
CDVM$ END REGION
CDVM$ GET_ACTUAL(eps)
```

Программа для кластера с ускорителями

Времена выполнения программы в секундах при использовании нескольких графических процессоров nVidia Fermi M2090 (k10.kiam.ru)



Времена выполнения программы в секундах при одновременном использовании графических процессоров и ядер ЦПУ

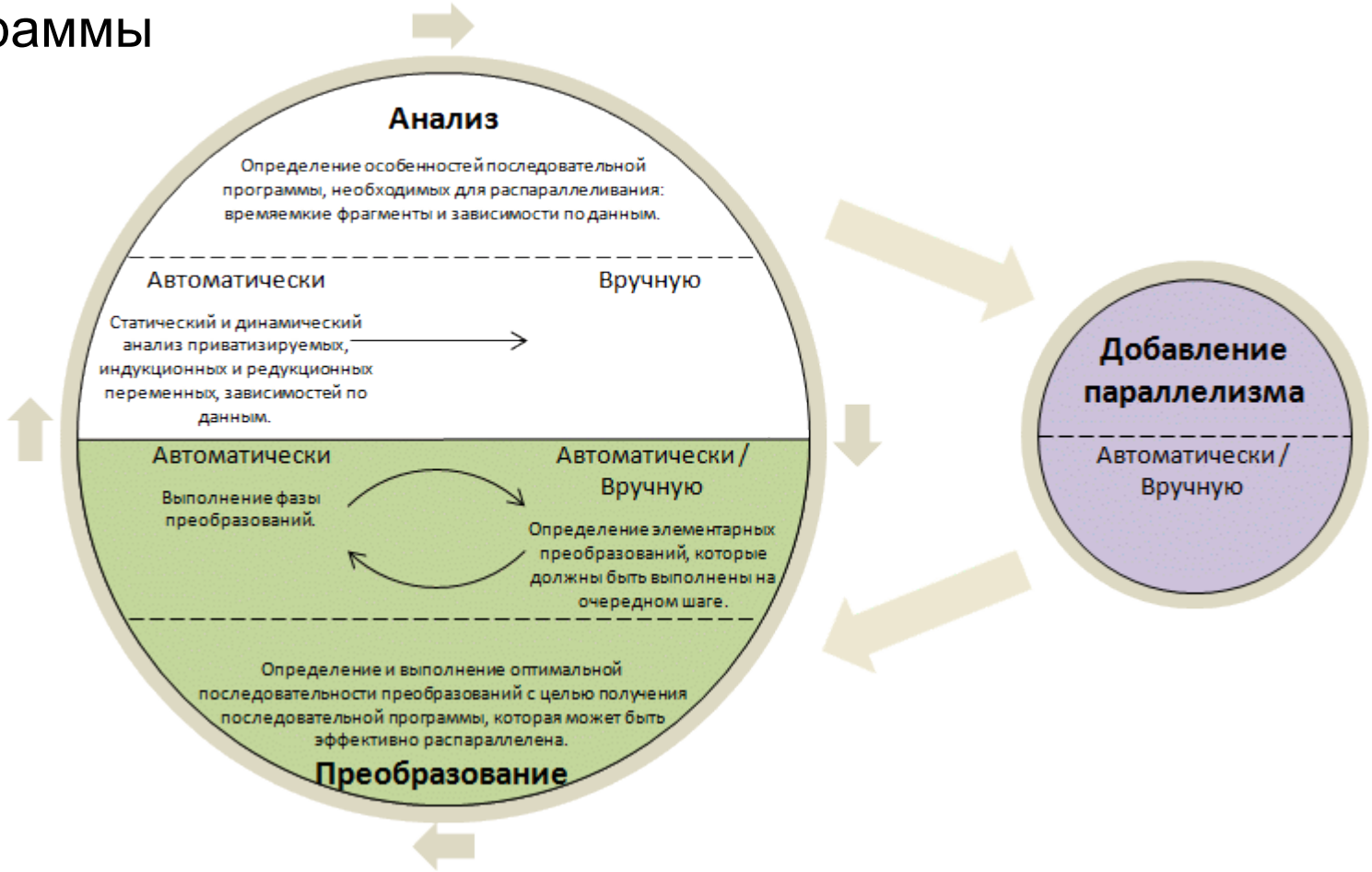


DVMH_SCHED_TECH=1
DVMH_CPU_PERF='0.08'
DVMH_CUDAS_PERF='0.46'

DVMH_SCHED_TECH=1
DVMH_CPU_PERF='0.095'
DVMH_CUDAS_PERF='0.905'

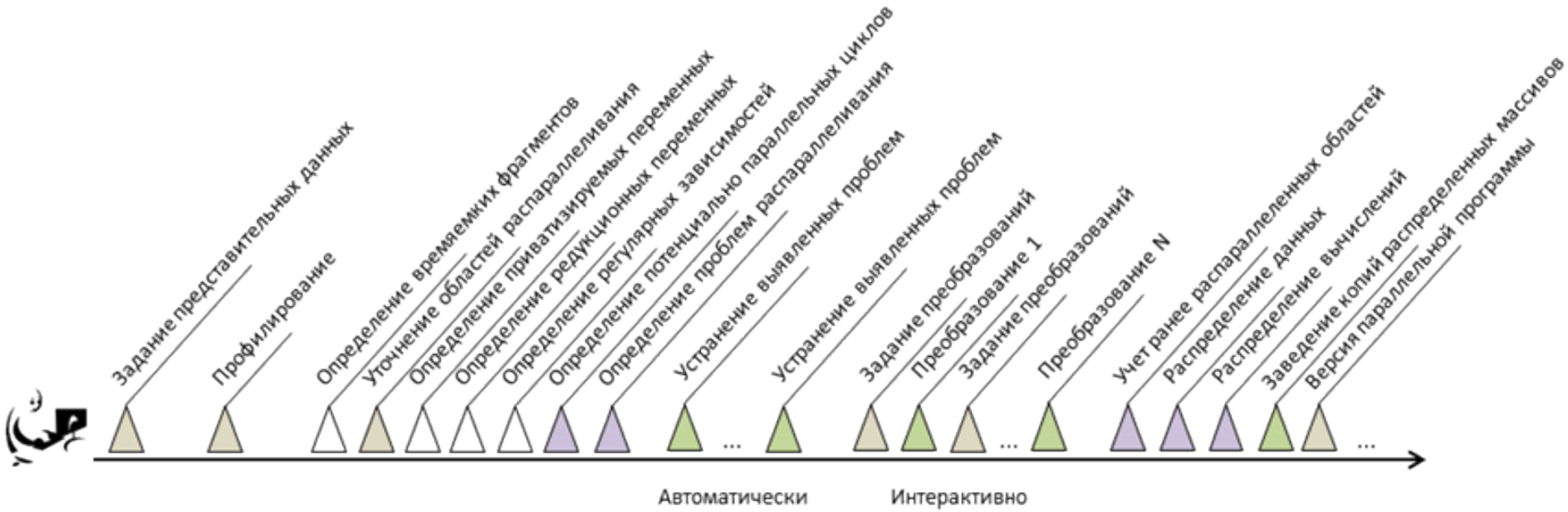
Система САПФОР

- Помогает программисту эффективно отображать его программы на многоядерные кластеры с ускорителями
- Взаимодействует с программистом в терминах последовательной программы



Развитие системы САПФОР

- Введены области распараллеливания для обеспечения инкрементального распараллеливания
- Распараллеливание программы представляет собой последовательность проходов анализа и преобразований
- Расширен класс распараллеливаемых программ за счет пересмотра алгоритмов распределения данных и вычислений



Использование САПФОР и DVM-системы

Разработаны параллельные программы для решения следующих прикладных задач:

- «Перколяция» (моделирование динамических процессов фильтрации в перколяционных решетках);
- «Композит» (моделирование многокомпонентной многофазной изотермической фильтрации при разработке месторождений нефти и газа);
- пакет прикладных программ «РЕАКТОР» (нейтронно-физический расчет ядерных реакторов и гибридных ядерных установок);
- Elasticity3D (численное моделирование 3D сейсмических полей в упругих средах для областей со сложной геометрией свободной поверхности);
- HyperbolicSolver2D (решение по явной схеме систем гиперболических уравнений в двумерных областях сложной формы с использованием неструктурированных сеток);

Использование САПФОР и DVM-системы

Разработаны параллельные программы для решения следующих прикладных задач:

- «Перколяция» (моделирование динамических процессов фильтрации в перколяционных решетках);
- «Композит» (моделирование многокомпонентной многофазной изотермической фильтрации при разработке месторождений нефти и газа);
- пакет прикладных программ «РЕАКТОР» (нейтронно-физический расчет ядерных реакторов и гибридных ядерных установок);
- Elasticity3D (численное моделирование 3D сейсмических полей в упругих средах для областей со сложной геометрией свободной поверхности);
- HyperbolicSolver2D (решение по явной схеме систем гиперболических уравнений в двумерных областях сложной формы с использованием неструктурированных сеток);

Использование САПФОР и DVM-системы

Разработаны параллельные программы для решения следующих прикладных задач:

- «Спекание 2D» и «Спекание 3D» (моделирование процессов плавления многокомпонентых порошков при селективном лазерном спекании на основе многокомпонентной и многофазной гидродинамической модели);
- «Кристаллизация 2D» и «Кристаллизация 3D» (моделирование процессов объёмной кристаллизации при воздействии на образец лазерного или электронного пучка на основе многокомпонентной и многофазной гидродинамической модели);
- QuantumBitStates (расчёт состояний кубитов квантового компьютера на основе решения нестационарного уравнения Шредингера для двух частиц с учетом спина);
- «Теплопроводность» (решение краевой задачи для двумерного квазилинейного параболического уравнения, записанного в дивергентной форме в различной постановке на неструктурированных треугольных сетках);

Использование САПФОР и DVM-системы

Разработаны параллельные программы для решения следующих прикладных задач:

- CONVD (трехмерная магнитная газодинамика для расчета солнечной конвекции с использованием реалистичной физической модели);
- MHPDV (моделирование сферического взрыва во внешнем магнитном поле с помощью решения уравнений идеальной магнитной гидродинамики);
- «Каверна» (моделирование циркуляционного течения в плоской квадратной каверне с движущейся верхней крышкой);
- «Контейнер» (моделирование течения вязкой тяжелой жидкости под действием силы тяжести в прямоугольном контейнере с открытой верхней стенкой и отверстием в одной из боковых стенок);
- …

Программа HyperbolicSolver2D

- Решение по явной схеме систем гиперболических уравнений (в основном, газовой динамики) в двумерных областях сложной формы с использованием неструктурированных сеток
- Часть большого развитого комплекса вычислительных программ (В.А. Гасилов, А.С. Болдарев)
- Для обеспечения максимальной универсальности и простоты дальнейшего развития код был написан на C++ с очень широким использованием объектно-ориентированного подхода
  - шаблоны, полиморфизм, и т.п.
  - богатая платформа базовых понятий и структур данных
  - 39000 LOC

Дополнительное распараллеливание MPI-программ

- Реализован новый режим работы DVM-системы - локально в каждом процессе
- Добавлена конструкция нераспределенного параллельного цикла
- Поэтапное распараллеливание и быстрая оценка эффективности DVMH-распараллеливания по ядрам ЦПУ и ГПУ перед проведением полноценного распараллеливания
- Возможность использовать DVMH-распараллеливание внутри узла кластера в готовых MPI-программах

Алгоритм Якоби в модели DVMH

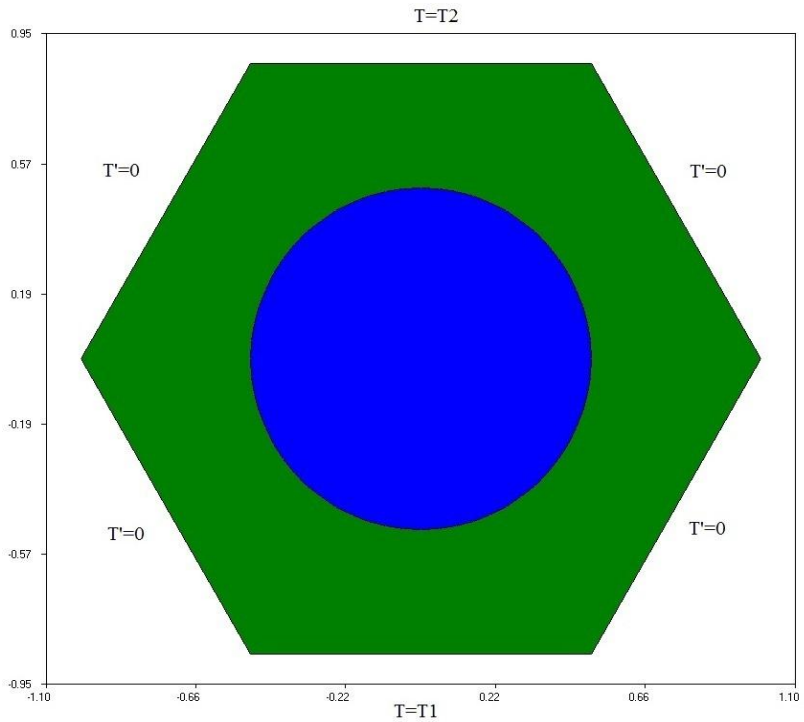
```
FILE *f;
double A[L][L];
double B[L][L];
int main(int argc, char *argv[]) {
   for(int it = 0; it < ITMAX; it++) {
      #pragma dvm region
      {
         #pragma dvm parallel(2)
         for (int i = 1; i < L - 1; i++)
            for (int j = 1; j < L-1; j++)  A[i][j] = B[i][j];
         #pragma dvm parallel(2)
         for (int i = 1; i < L - 1; i++)
            for (int j = 1; j < L - 1; j++)
               B[i][j] = (A[i - 1][j] + A[i + 1][j] + A[i][j - 1] + A[i][j + 1]) / 4.;
      }
   }
  f = fopen("jacobi.dat", "wb");
  #pragma dvm get_actual(B)
  fwrite(B, sizeof(double), L * L, f);
  fclose(f);
  return 0;
}
```

Программа HyperbolicSolver2D

- Решение по явной схеме систем гиперболических уравнений (в основном, газовой динамики) в двумерных областях сложной формы с использованием неструктурированных сеток
- Для обеспечения максимальной универсальности и простоты дальнейшего развития код был написан на C++ с очень широким использованием объектно-ориентированного подхода
  - шаблоны, полиморфизм, и т.п.
  - богатая платформа базовых понятий и структур данных
  - 39000 LOC
- Модификации только локальные, в одном модуле в 3 тыс.строк, сводятся к добавлению нескольких (около 10) директив
- Получено ускорение
  - 12 ядер ЦПУ - 9,83 раза
  - ГПУ NVIDIA GTX TITAN - 18 раз

Двумерная задача теплопроводности в шестиграннике

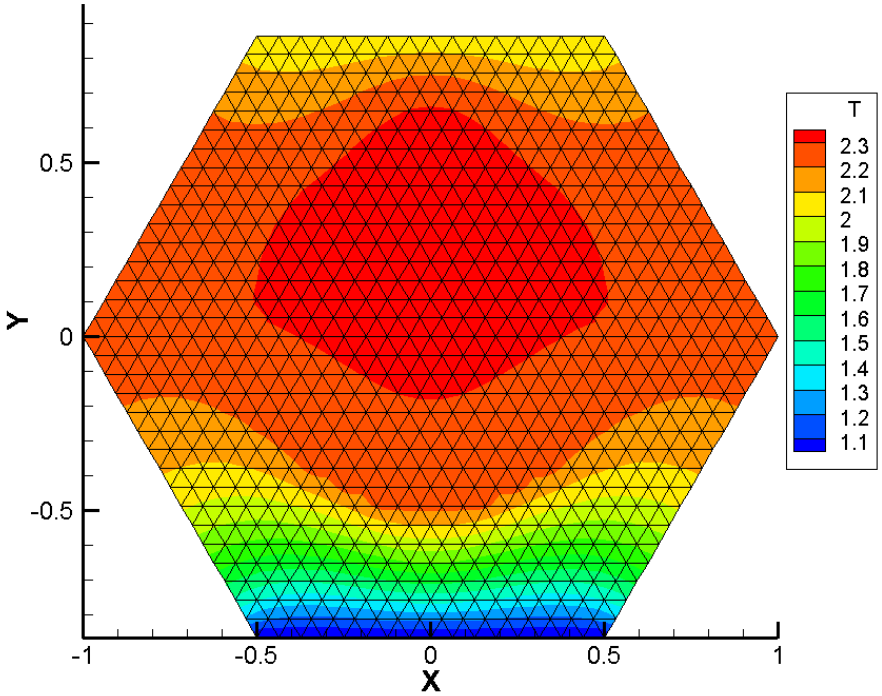
Область состоит из двух материалов с различными коэффициентами температуропроводности.




```
do i = 1, np2
  nn = ii(i)
  nb = npa(i)
  if (nb.ge.0) then
    s1 = FS(xp2(i),yp2(i),tv)
    s2 = 0d0
    do j = 1, nn
      j1 = jj(j,i)
      s2 = s2 + aa(j,i) * tt1(j1)
    enddo
    s0 = s1 + s2
    tt2(i) = tt1(i) + tau * s0
  else if (nb.eq.-1) then
    tt2(i) = vtemp1
  else if (nb.eq.-2) then
    tt2(i) = vtemp2
  endif
  s0 = (tt2(i) - tt1(i)) / tau
  gt = DMAX1(gt,DABS(s0))
enddo
do i = 1, np2
  tt1(i) = tt2(i)
enddo
```

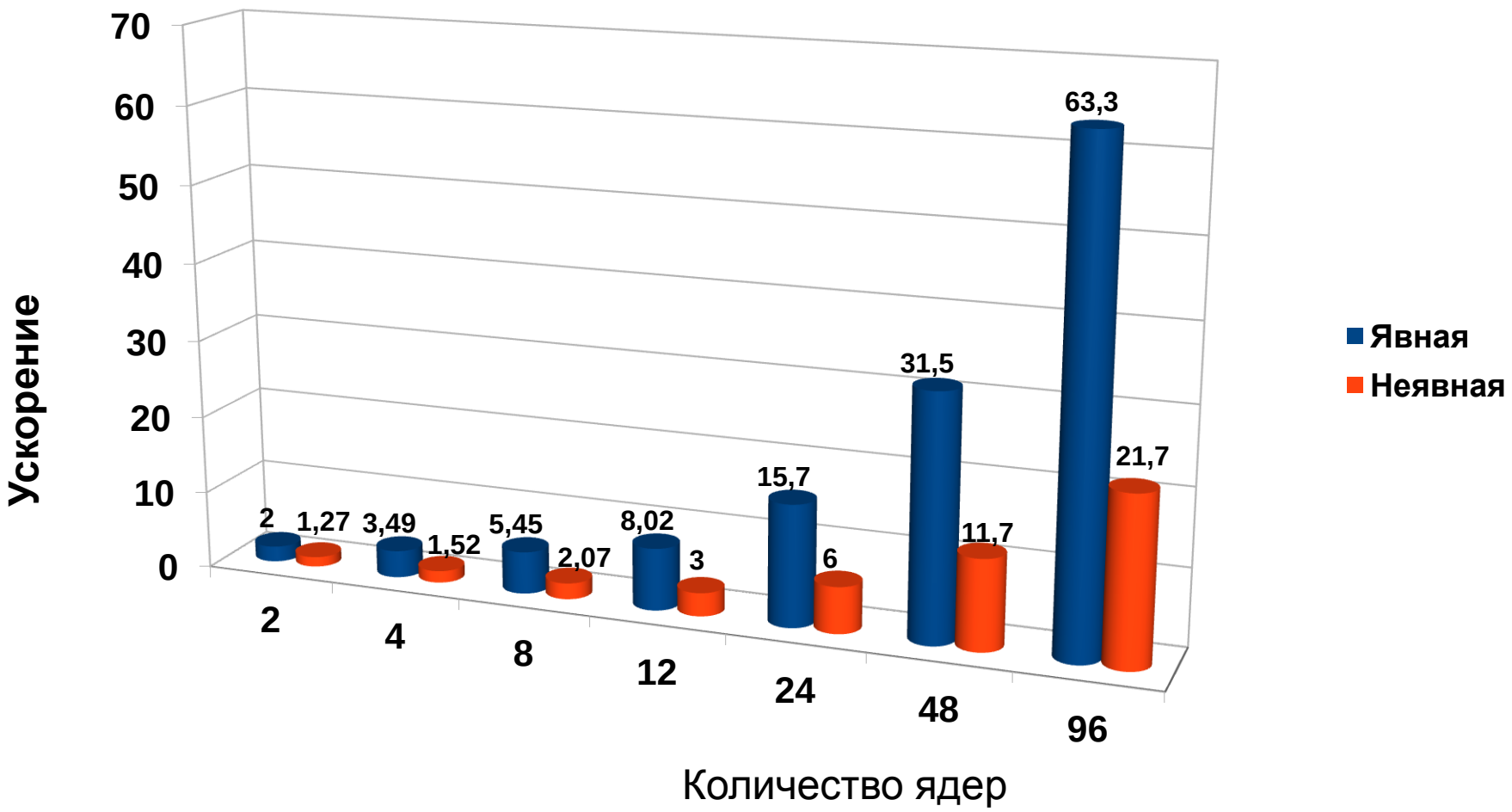
Двумерная задача теплопроводности в шестиграннике

Стационарное распределение температуры

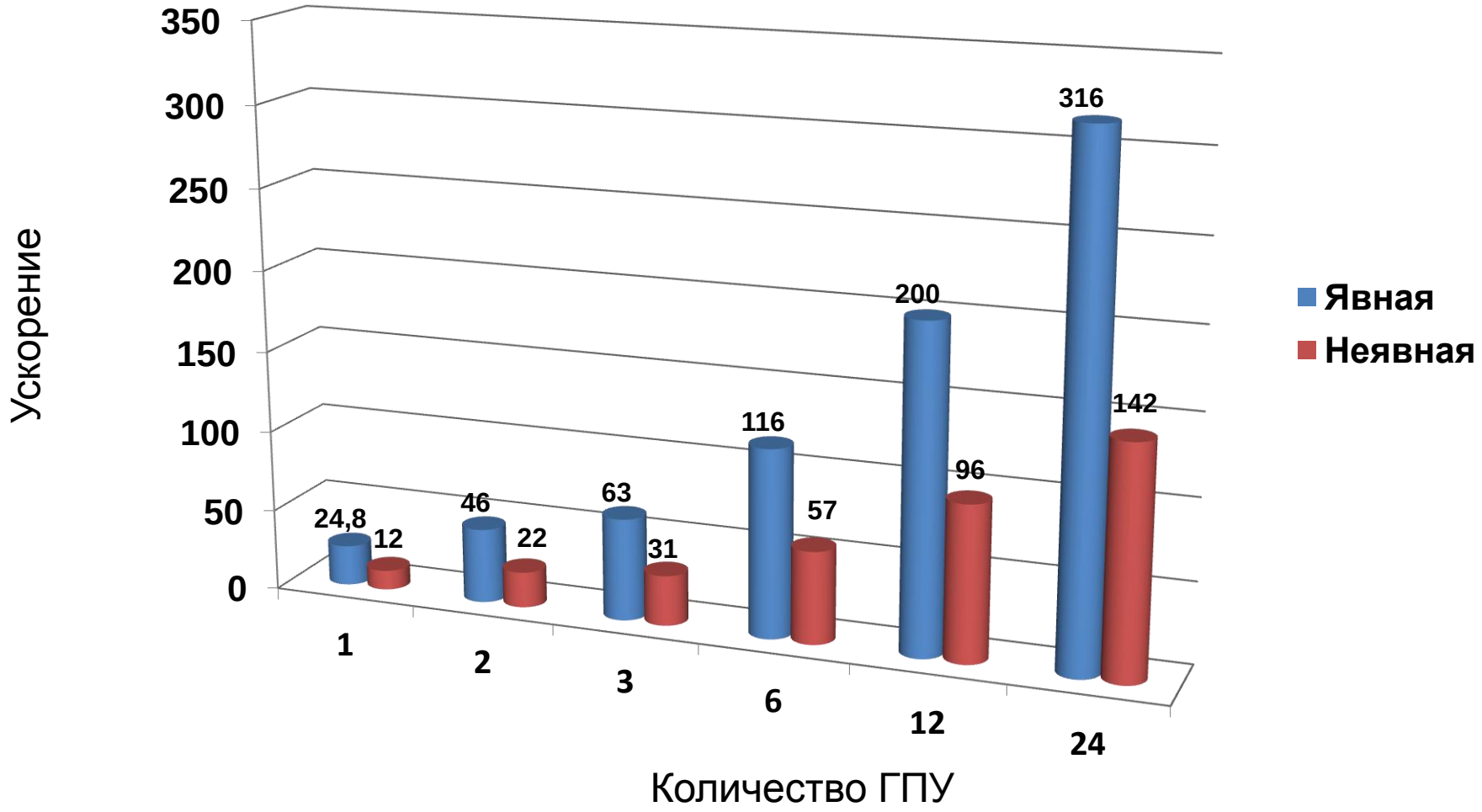



```
do i = 1, np2
  nn = ii(i)
  nb = npa(i)
  if (nb.ge.0) then
    s1 = FS(xp2(i),yp2(i),tv)
    s2 = 0d0
    do j = 1, nn
      j1 = jj(j,i)
      s2 = s2 + aa(j,i) * tt1(j1)
    enddo
    s0 = s1 + s2
    tt2(i) = tt1(i) + tau * s0
  else if (nb.eq.-1) then
    tt2(i) = vtemp1
  else if (nb.eq.-2) then
    tt2(i) = vtemp2
  endif
  s0 = (tt2(i) - tt1(i)) / tau
  gt = DMAX1(gt,DABS(s0))
enddo
do i = 1, np2
  tt1(i) = tt2(i)
enddo
```

Ускорение на ЦПУ для сетки 8 млн. узлов (k100.kiam.ru)



Ускорение на ГПУ (Tesla C2050) для сетки 8 млн. узлов (k100.kiam.ru)



Новые возможности для работы с нерегулярными сетками

В 2016 году сформулированы предложения по расширению DVMH-модели:

- Задание произвольных поэлементных распределений, в том числе получаемые пакетами Metis, Chaco,…
- Построение согласованных распределений на основе имеющихся (блочных или поэлементных)
- Задание произвольных по содержанию буферов удаленных элементов с эффективным однородным доступом к ним и обновлением
- Возможность реорганизации данных - оптимизации шаблона доступа к памяти путем изменения порядка хранения локальных элементов
- Сохранение быстрого доступа к распределенным массивам с помощью механизмов перехода на локальную индексацию

Новые правила распределения

- Косвенное - задается массивом целых чисел, размер которого равен размеру косвенно распределяемого измерения, а значения задают номер домена.
  distribute A[indirect(B)]
- Производное - задается правилом, по форме похожим на правило выравнивания (ALIGN) модели DVMH: возможность согласованно распределять сеточные элементы. Например, ячейки, ребра, вершины.
  distribute A[derived([cells[i][0]: cells[i][2]] with cells[@i])]

Новые теневые грани

- Теневые грани - это набор элементов, не принадлежащих текущему процессу, для которых:
  - Возможен доступ без специальных указаний из любой точки программы
  - Имеются специальные средства работы с ними: *shadow_renew, shadow_compute, across*
  - Любой набор удаленных элементов задается аналогично производному распределению *shadow_add(nodes[neigh[i][0]:neigh[i][numneigh[i]-1] with nodes[@i]] = neighbours)*

Переход к локальной индексации

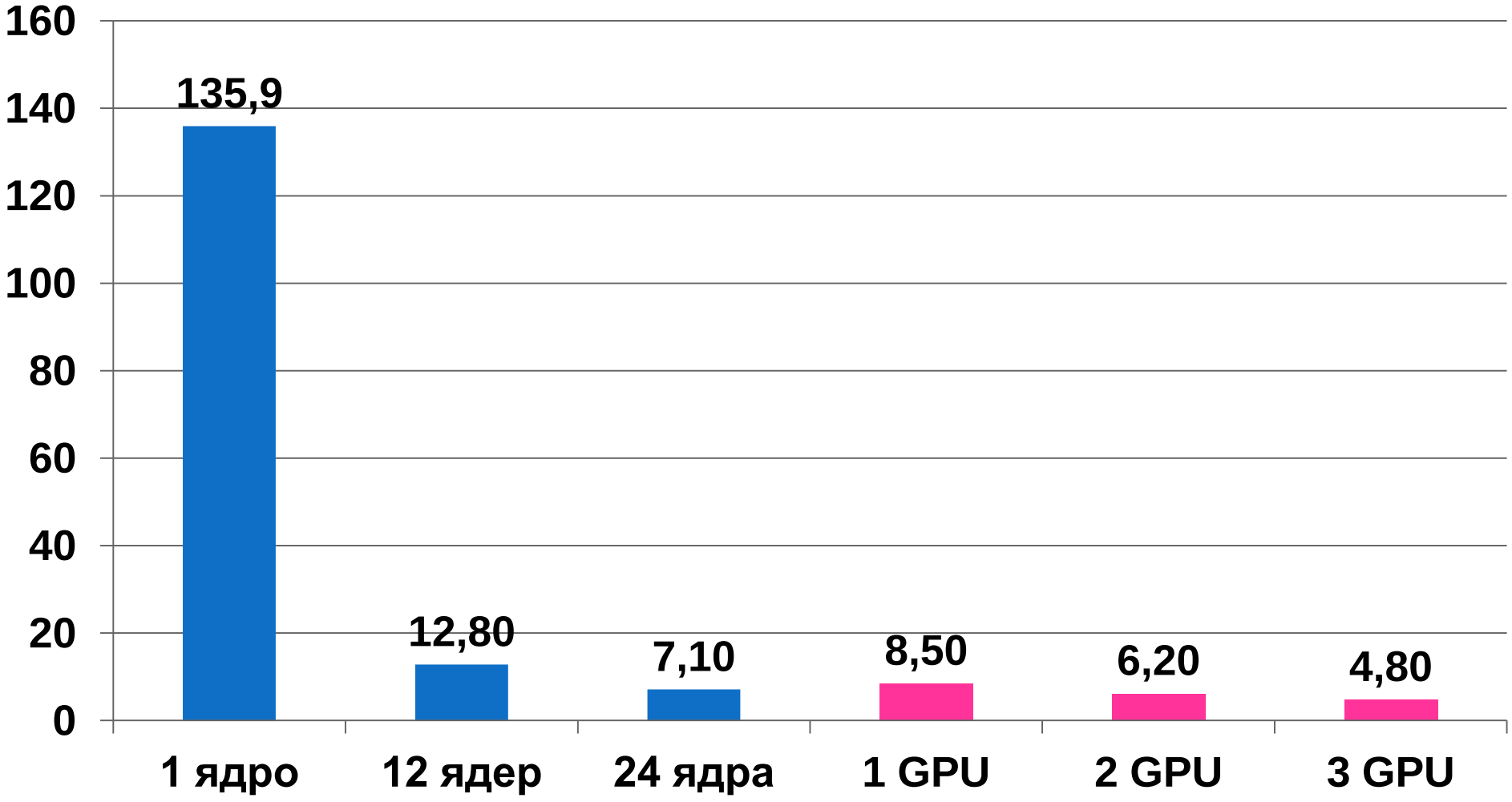
- Процедура для перевода глобального (исходного) индекса в локальный (непосредственно для доступа к памяти) слишком долгая
- Для блочных распределений они совпадают
- Для поэлементных временно введена исполняемая директива локализации значений индексных массивов

  localize(neigh => nodes[:])
- В результате дальнейшие действия производятся над локальными индексами и нет необходимости менять компиляцию исполняемых конструкций

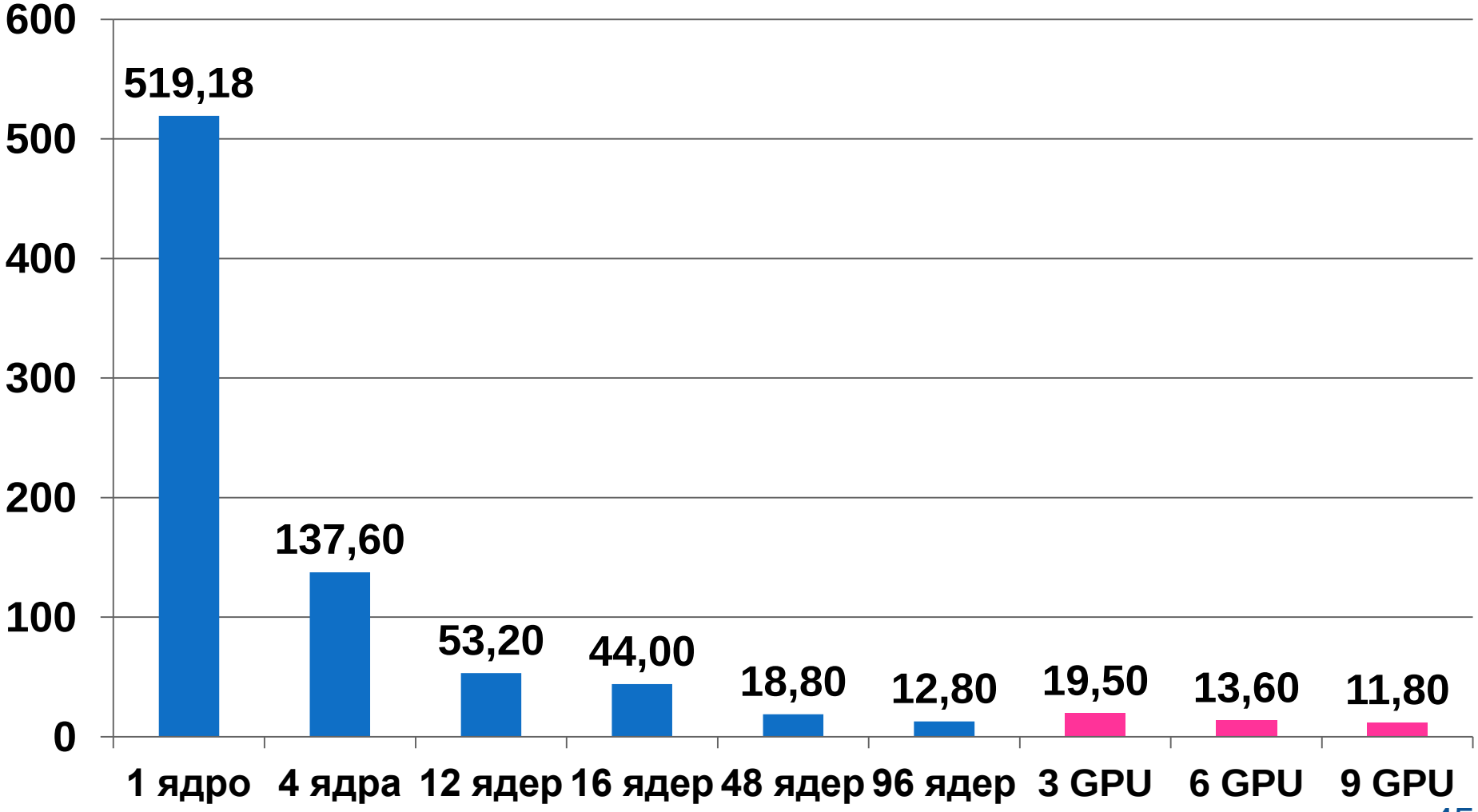
Программа для решения задачи газовой динамики методом Галеркина

- Для решения задач газовой динамики в настоящее время широко применяется метод Галеркина с разрывными базисными функциями, который характеризуется высоким порядком точности получаемого решения
- Последовательная версия программы разработана в ИПМ им. М.В. Келдыша РАН (Ладонкина М.Е., Тишкин В.Ф.)
- В качестве тестовой задачи рассматривается простая волна, в которой энтропия и инвариант Римана являются постоянными, используется классический лимитер Кокбурна, который легко реализуется в многомерном случае на сетках произвольной структуры
- Задача рассматривается в двумерной постановке

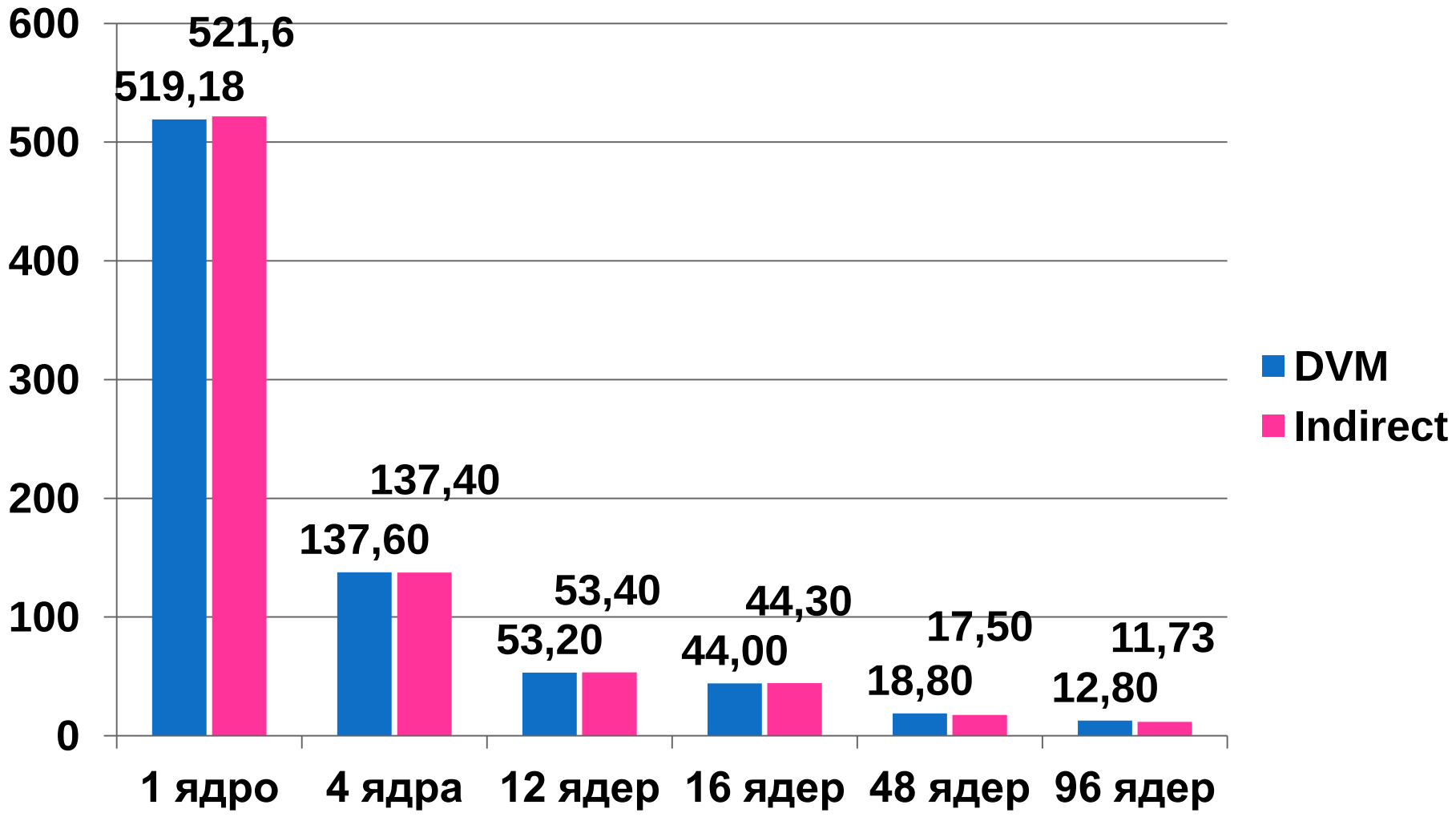
Время выполнения программы для решения задачи газовой динамики методом Галеркина для сетки из 60000 ячеек (k100.kiam.ru)



Время выполнения программы для решения задачи газовой динамики методом Галеркина для сетки из 200000 ячеек (k100.kiam.ru)



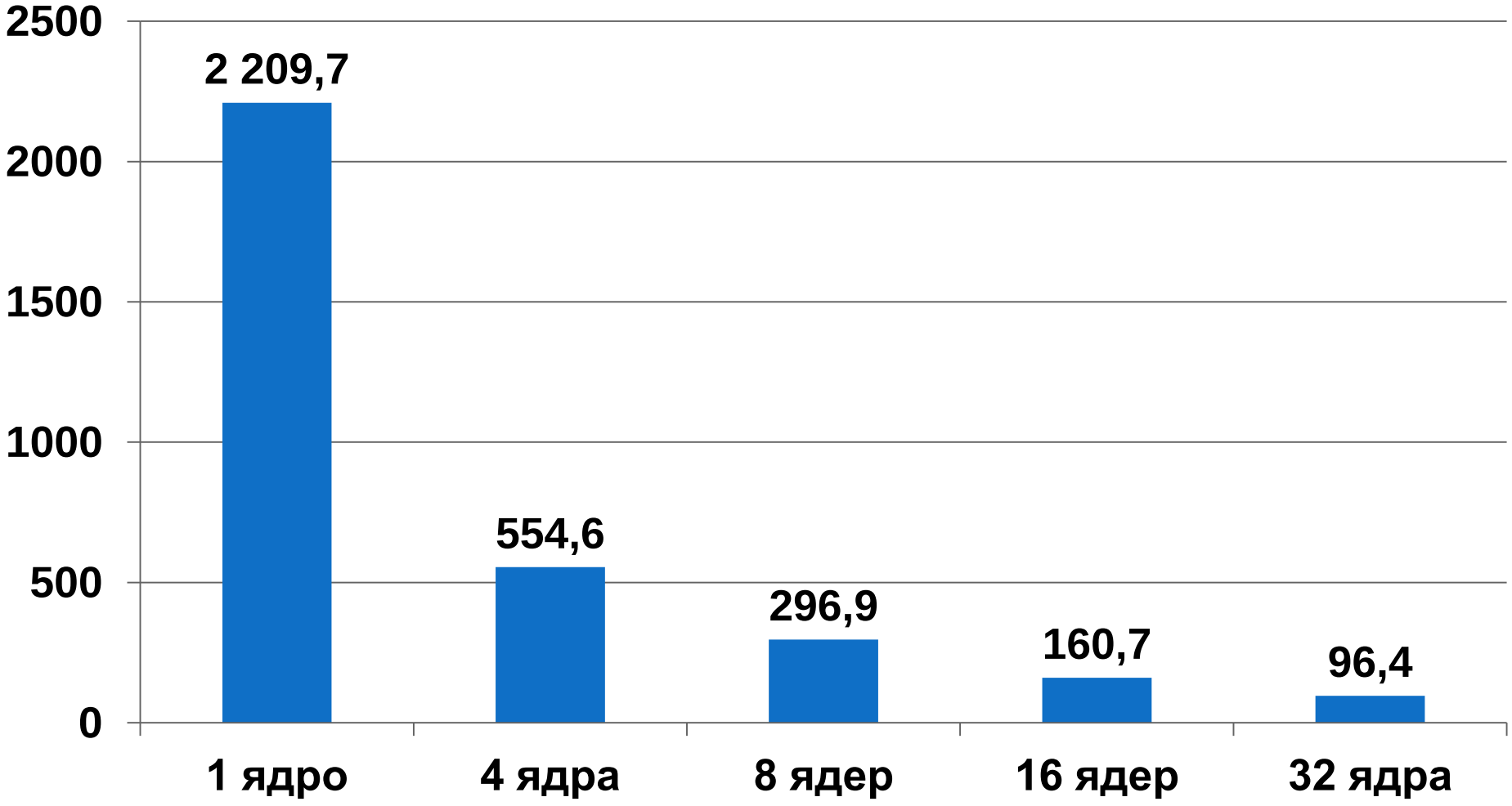
Сравнение времени выполнения программы для решения задачи газовой динамики при использовании расширения (сетка из 200000 ячеек)



Программа для расчета течений плазмы с учетом переноса излучения на основе РМГД модели

- На основе уравнений МГД с учетом электро-, теплопроводности и переноса излучения в ИПМ им. М.В. Келдыша РАН разработан программный комплекс для численного исследования динамики потоков плазмы в каналах квазистационарных плазменных ускорителей (Козлов А.Н., Коновалов В.С.)
- Использование данного программного комплекса позволяет определить условия для получения высокоэнергетических потоков плазмы, в которых энергия ионов на выходе из КСПУ является достаточно большой, чтобы обеспечить последующую реакцию синтеза в перспективных термоядерных установках

Время выполнения программы для расчета течений плазмы (k100.kiam.ru)



Выводы

- DVM-система автоматизирует процесс разработки параллельных программ
- Получаемые DVMH-программы без каких-либо изменений могут эффективно выполняться на кластерах различной архитектуры, использующих многоядерные универсальные процессоры и графические ускорители
- Это достигается за счет различных оптимизаций, которые выполняются как статически, при компиляции DVMH-программ, так и динамически. Получаемые параллельные программы могут настраиваться при запуске на выделенные для их выполнения ресурсы - количество узлов кластера, ядер, ускорителей и их производительность

http://dvm-system.org

Выводы

- Данные оптимизации позволяют добиться высокой эффективности выполнения различных тестовых программ и реальных приложений
- DVM-система активно развивается, появляются новые возможности, которые расширяют ее область применимости, позволяют рапараллеливать не только задачи на структурированных сетках (для которых DVM-система была предназначена изначально), но и задачи на неструктурированных сетках

http://dvm-system.org

Вопросы, замечания?

http://dvm-system.org