Опыт решения прикладных задач с использованием DVM-системы

В.Ф. Алексахин, **В.А. Бахтин**, Д.А. Захаров, А.С. Колганов, А.В. Королев, В.А. Крюков, Н.В. Поддерюгина, М.Н. Притула

dvm@keldysh.ru

Москва, 26 сентября, 2017

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
- Распараллеливание прикладных задач, использующих структурированные сетки
  - Многокомпонентная многофазная изотермическая фильтрация
- Распараллеливание прикладных задач, использующих неструктурированные сетки
  - Задача теплопроводности в шестиграннике
- Дополнительное распараллеливание MPI-программ

© ИПМ ИМ. М.В. КЕЛДЫША РАН

```
      PROGRAM   JACOBY_DVMH
      PARAMETER   (L=4096,  ITMAX=100)
      REAL    A(L,L), B(L,L)
!DVM$ DISTRIBUTE   ( BLOCK,  BLOCK)  ::  A
!DVM$ ALIGN  B(I,J)  WITH  A(I,J)
      PRINT *,  '**********  TEST_JACOBI  **********'
      DO IT  =  1,  ITMAX
!DVM$     REGION INOUT(A,B)
!DVM$       PARALLEL (J,I) ON  A(I,  J)
            DO  J  =  2, L-1
               DO  I  =  2, L-1
                  A(I, J)  =  B(I, J)
               ENDDO
            ENDDO
!DVM$       PARALLEL (J,I) ON B(I,  J), SHADOW_RENEW (A)
            DO  J = 2,  L-1
               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
!DVM$     END REGION
          ENDDO
!DVM$ IO_MODE(LOCAL, ASYNCHRONOUS)
      OPEN(3, FILE='JAC.DAT', ACCESS='STREAM')
!DVM$ GET_ACTUAL(B)
      WRITE(3, *) B
      CLOSE(3)
      END
```

Алгоритм Якоби в модели DVMH

© ИПМ ИМ. М.В. КЕЛДЫША РАН

Алгоритм Якоби в модели DVMH

```
FILE *f;
#pragma dvm array distribute[block][block] shadow[1:1][1:1]
double A[L][L];
#pragma dvm array align ([i][j] with A[i][j])
double B[L][L];
int main(int argc, char *argv[]) {
   for(int it = 0; it < ITMAX; it++) {
      #pragma dvm region inout (A,B)
      {
         #pragma dvm parallel([i][j] on A[i][j])
         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([i][j] on B[i][j]), shadow_renew(A)
         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", "wbsl");
   #pragma dvm get_actual(B)
   fwrite(B, sizeof(double), L * L, f);
   fclose(f);
   return 0;
}
```


© ИПМ ИМ. М.В. КЕЛДЫША РАН

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

© ИПМ ИМ. М.В. КЕЛДЫША РАН

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

© ИПМ ИМ. М.В. КЕЛДЫША РАН

Состав DVM-системы

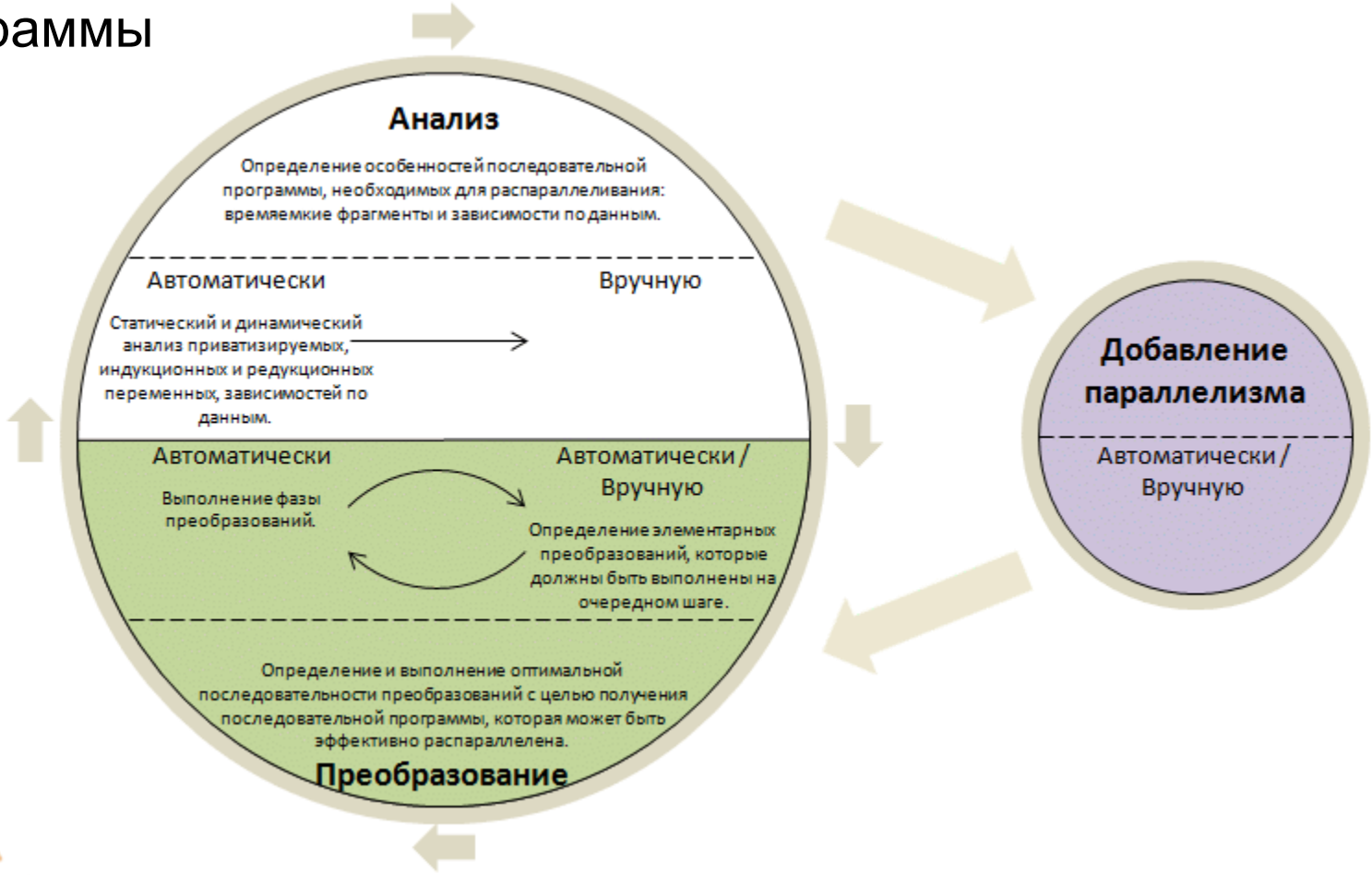
Система состоит из следующих компонент:

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

© ИПМ ИМ. М.В. КЕЛДЫША РАН

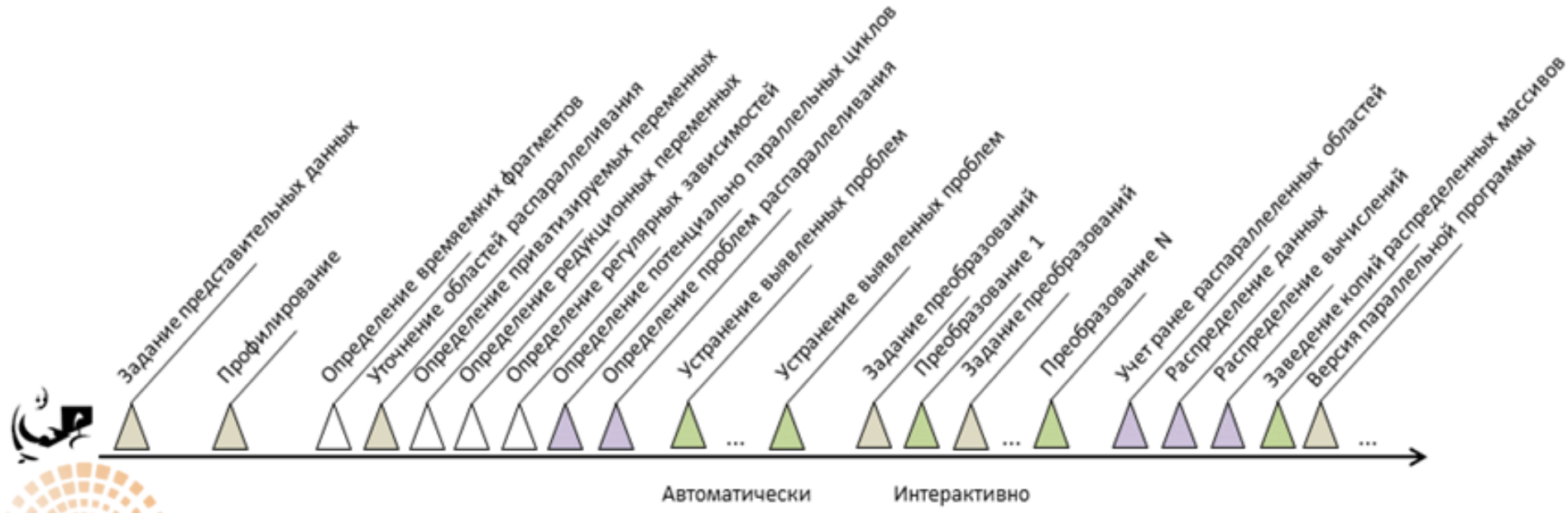
Система САПФОР

- 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



© ИПМ ИМ. М.В. КЕЛДЫША РАН

САПФОР. Инкрементальное распараллеливание

- 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 или без сохранения этих решений
- Возможность ручного распараллеливания некоторых фрагментов и учета принятых программистом решений при распараллеливании других фрагментов

© ИПМ ИМ. М.В. КЕЛДЫША РАН

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);

© ИПМ ИМ. М.В. КЕЛДЫША РАН

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);
- …

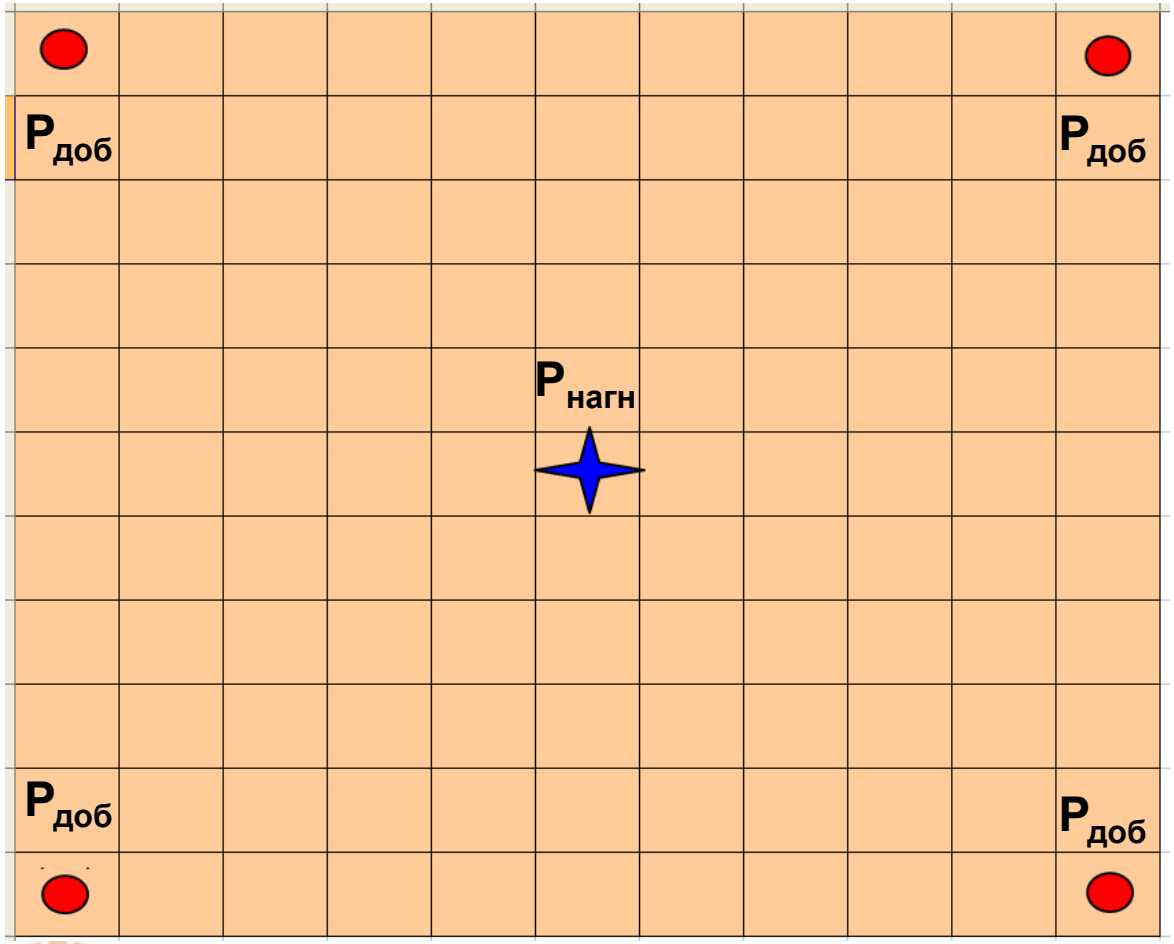
© ИПМ ИМ. М.В. КЕЛДЫША РАН

Многокомпонентная многофазная изотермическая фильтрация

- Композиционные модели фильтрации используются при подробном моделировании залежей, содержащих легкие углеводороды (конденсат и газ) в том случае, когда необходимо тщательно описывать массообмен между фазами, либо когда пластовые флюиды содержат ценные неуглеводородные компоненты
- Часто используются для изучения методов увеличения нефтеотдачи при закачке газов высокого давления, азота, углекислого газа и других агентов
- Последовательная версия программы (далее Композит) разработана в НИИСИ РАН

© ИПМ ИМ. М.В. КЕЛДЫША РАН

Тестовые расчеты с закачкой в пласт сухого и жирного газа (элемент 5-точечной системы разработки)



$P_{нач}$ = 244 атм

$P_{доб}$ = 244 атм

$P_{нагн}$ = 400 атм

Схема неявная по давлению и явная по концентрации компонентов

© ИПМ ИМ. М.В. КЕЛДЫША РАН

Распараллеливание программы Композит

- Перенос программы в ОС Linux. Отладка программы
- Определение времяемких фрагментов программы
- Распараллеливание времяемких фрагментов программы
- Добавление распараллеливаемых фрагментов с сохранением уже найденных решений

© ИПМ ИМ. М.В. КЕЛДЫША РАН

Сравнительная отладка программы

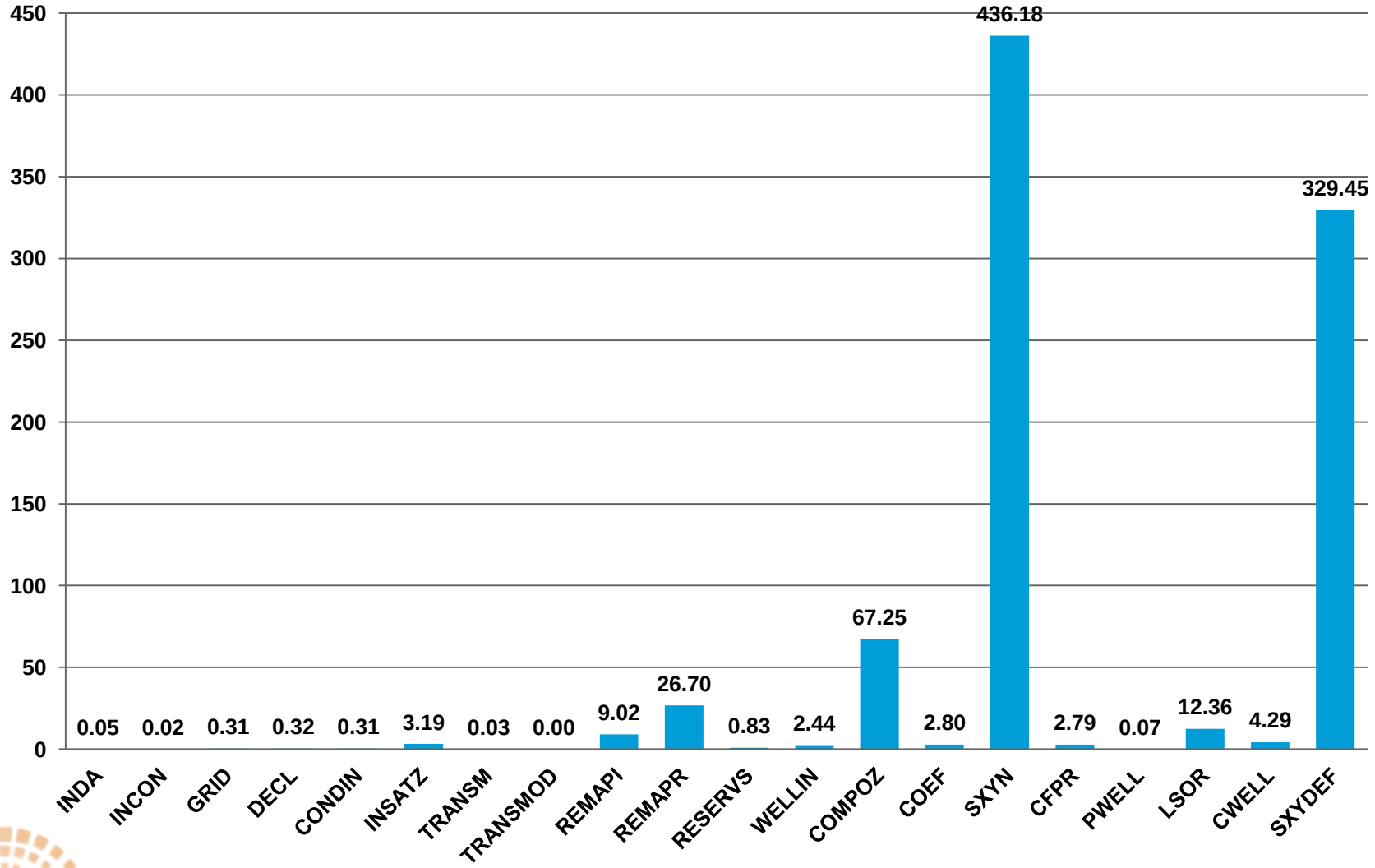
- Для поиска ошибок используется метод накопления и сравнения результатов вычислений, который позволяет определить место в программе и момент, когда появляются расхождения
- При трассировке вычислений выполняется сбор информации обо всех чтениях и модификациях переменных, о начале выполнения каждого 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

```
INTERVAL ( NLINE=463 SOURCE=COMPOZ.FOR ) LEVEL=3 USER EXE_COUNT=564
-- The main characteristics ---
Parallelization efficiency      1.0000
Execution time                 67.2498
Processors                        1
Total time                     67.2498
Productive time                67.2498 ( CPU= 67.2063  Sys= 0.0435 I/O= 0.0000 )
--- The comparative characteristics ---
                          Tmin   N proc    Tmax  N proc  Tmid
Execution time           67.2498     1   67.2498     1   67.2498
User CPU time            67.2063     1   67.2063     1   67.2063
Sys. CPU time             0.0435     1    0.0435     1    0.0435
Processors                   1       1         1     1         1
--- The execution characteristics ---
                          1
Execution time           67.2498
User CPU time            67.2063
Sys. CPU time             0.0435
Processors                   1
```


© ИПМ ИМ. М.В. КЕЛДЫША РАН

Времена выполнения различных процедур программы Композит



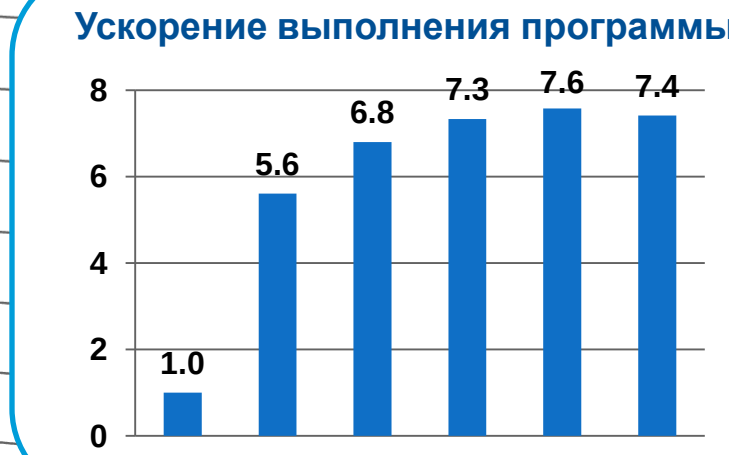
© ИПМ ИМ. М.В. КЕЛДЫША РАН

Распараллеливание времяемких фрагментов программы

- Анализ операторов (циклов) программы на предмет их возможного параллельного выполнения
- Определение наилучшего варианта распределения данных, требуемых для выполнения этих операторов
- Организация параллельных вычислений (расстановка DVMH-директив)
- Если распределяемые данные используются в других фрагментах программы, то требуется завести массивы-копии
- Организация копирования информации между исходными массивами и их копиями

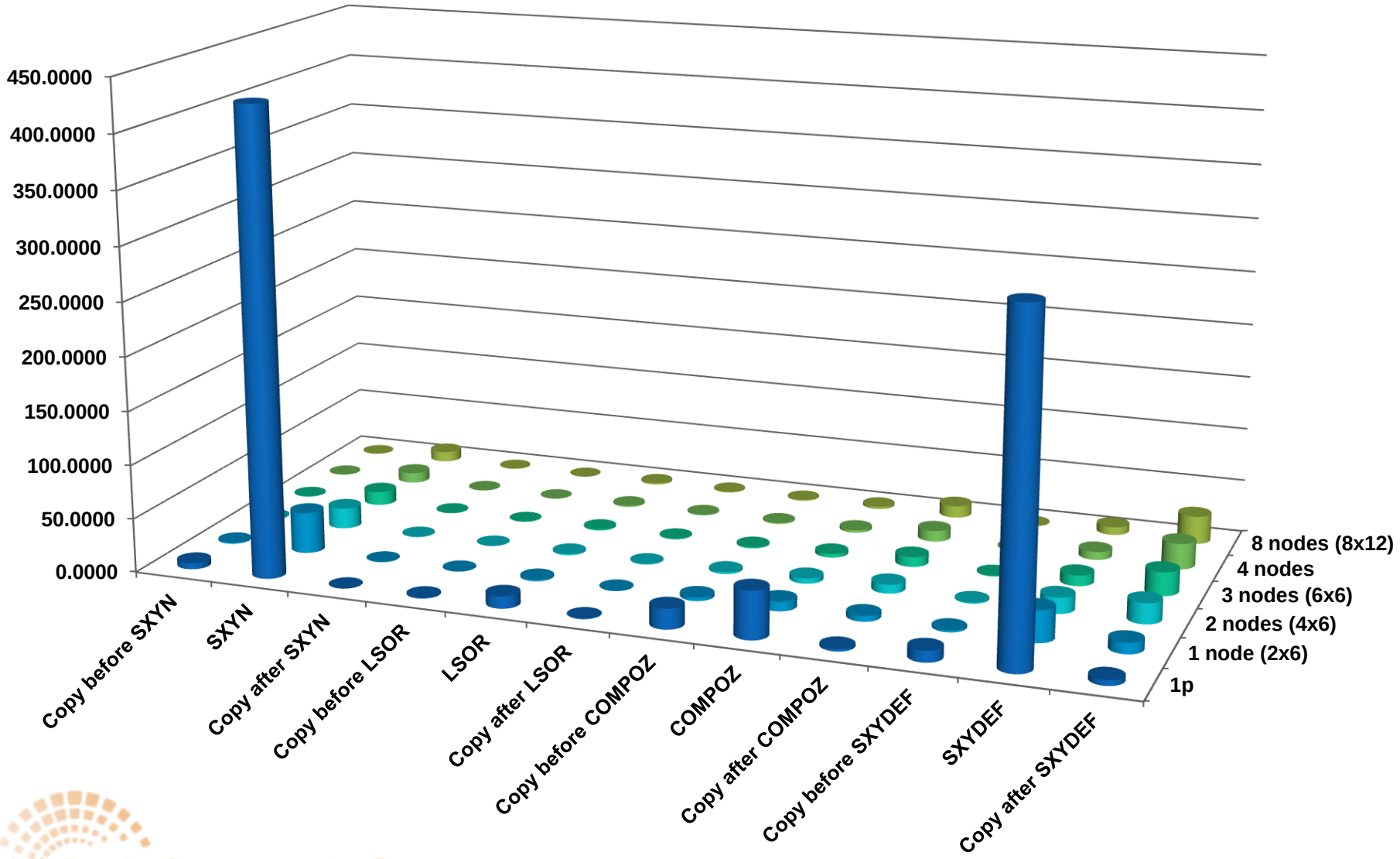
Времена выполнения частично распараллеленной программы на разном числе узлов K-100

На первом этапе были реализованы параллельные версии процедур COMPOZ, SXYN, SXYDEF, LSOR. (2700 из 13844 строк программы)

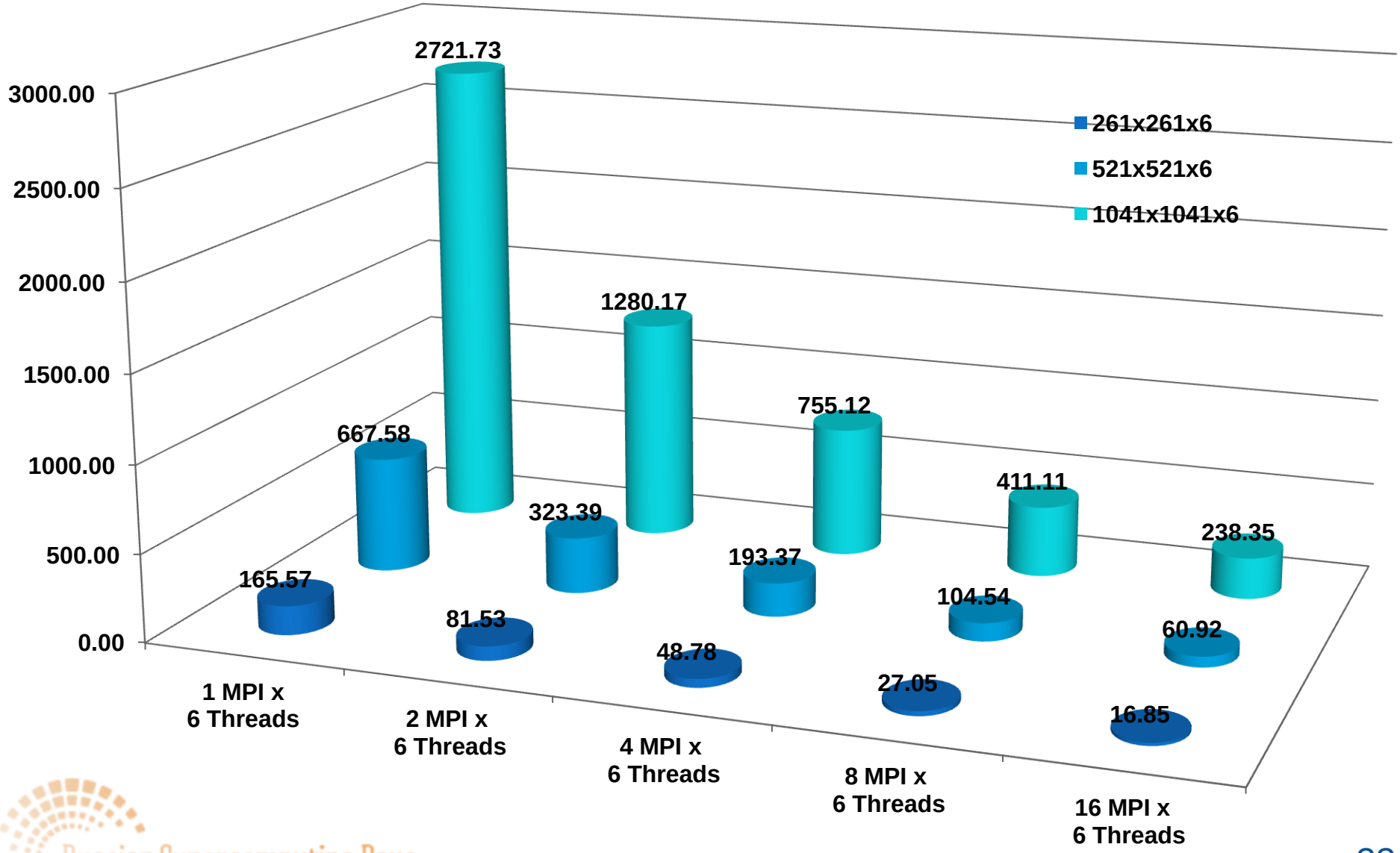


© ИПМ ИМ. М.В. КЕЛДЫША РАН

Накладные расходы на копирование данных между исходными массивами и их копиями

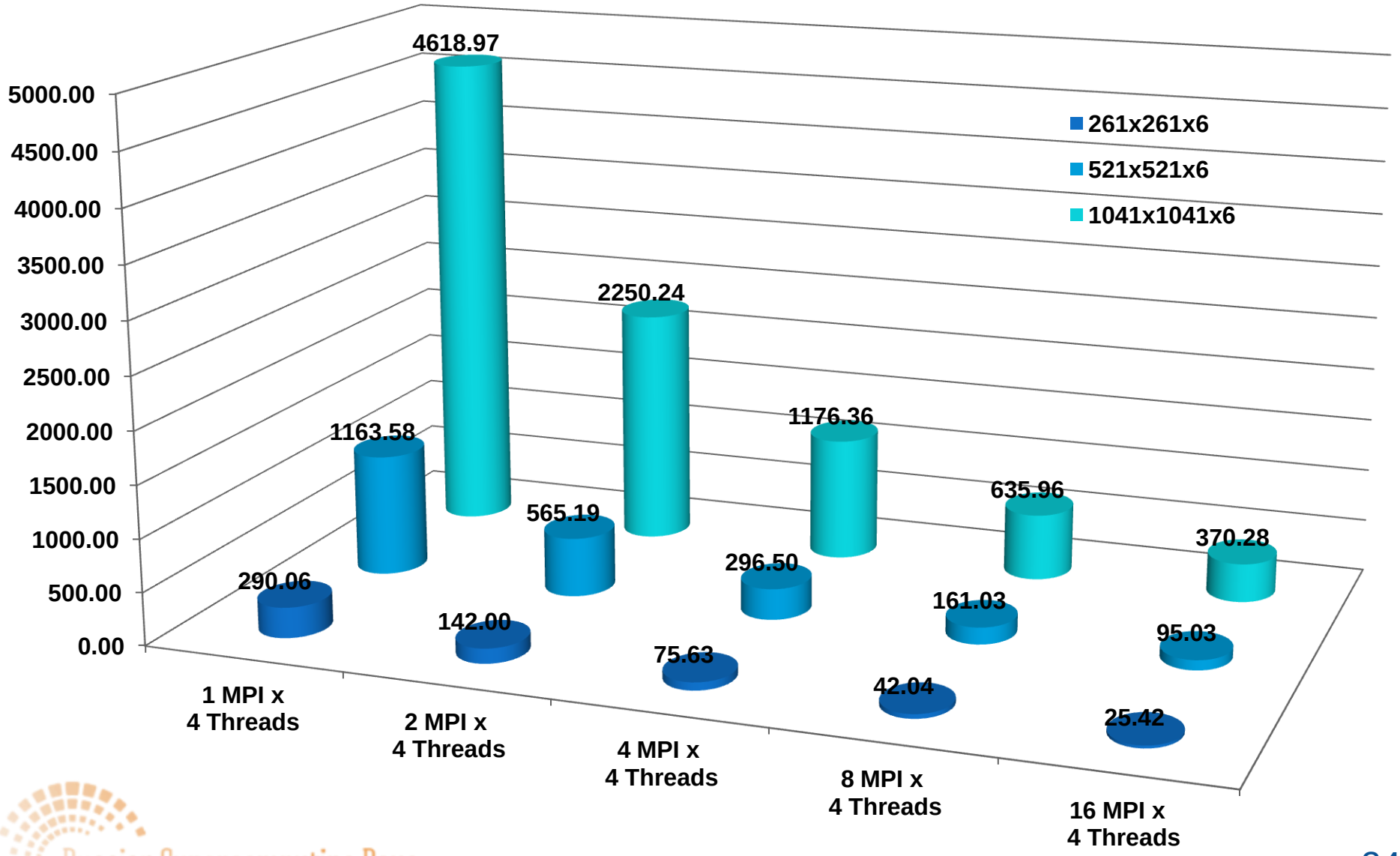


Времена выполнения 100 итераций программы Композит на суперкомпьютере К-100

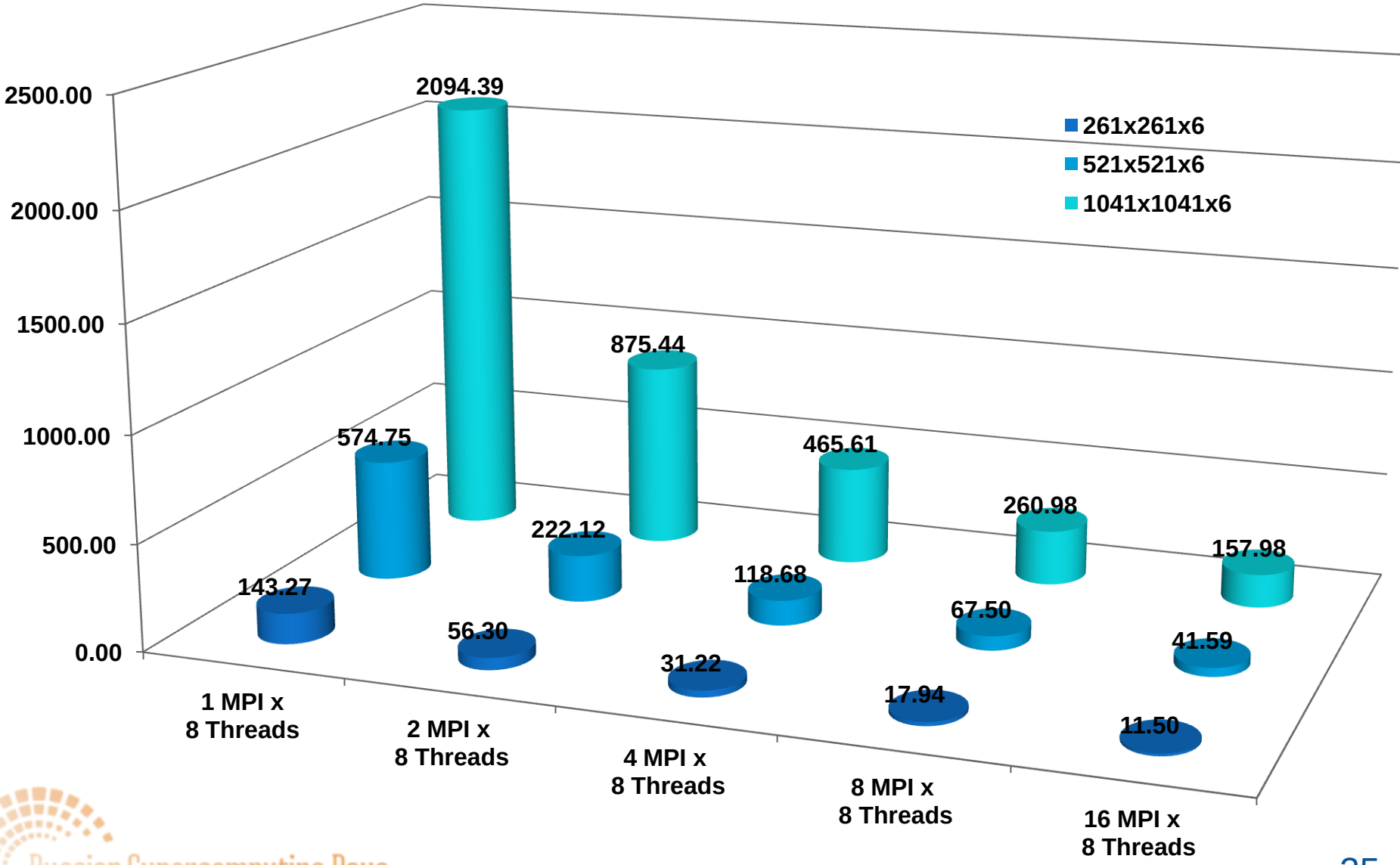


© ИПМ ИМ. М.В. КЕЛДЫША РАН

Времена выполнения 100 итераций программы Композит на суперкомпьютере Ломоносов

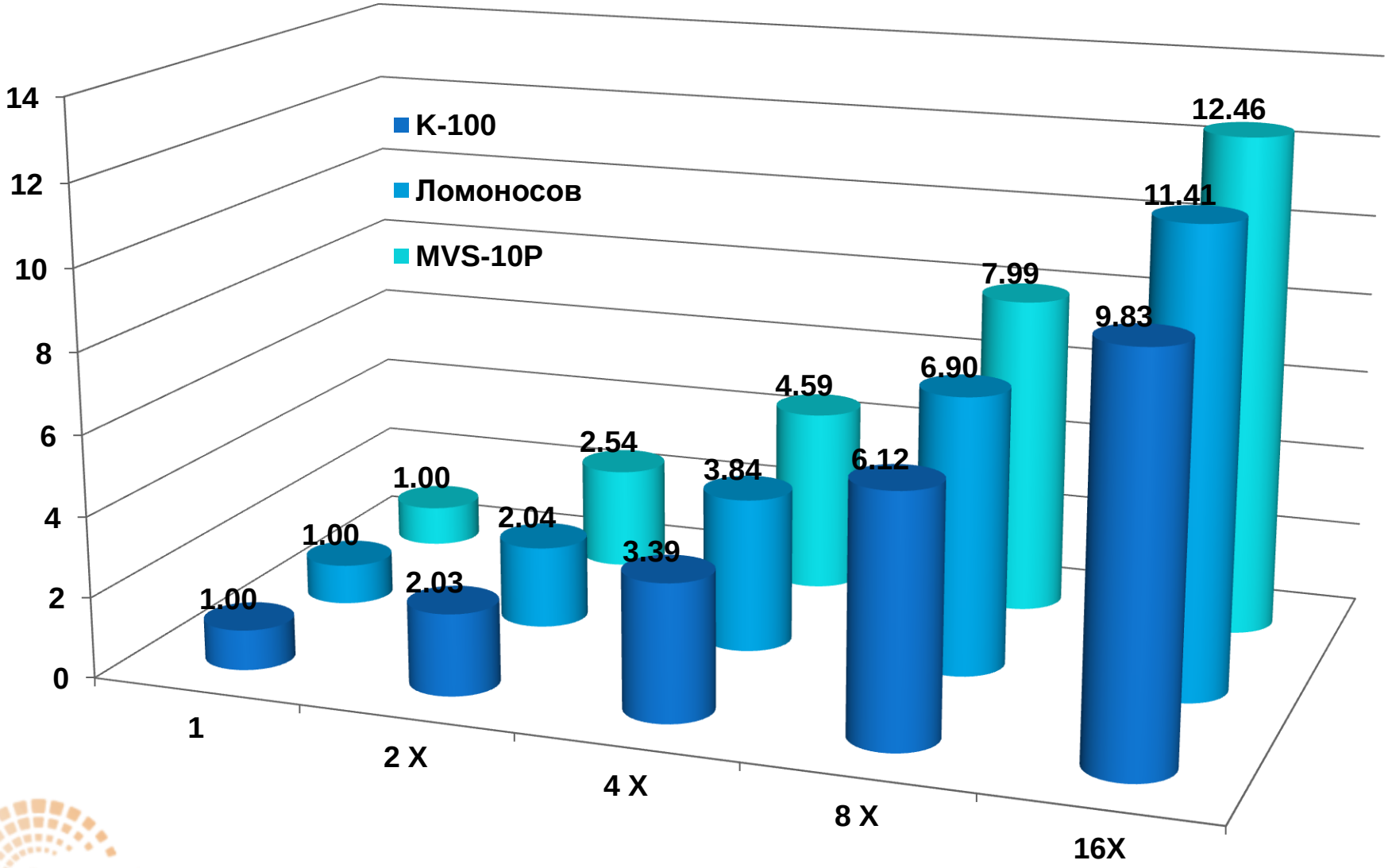


Времена выполнения 100 итераций программы Композит на суперкомпьютере MVS-10P



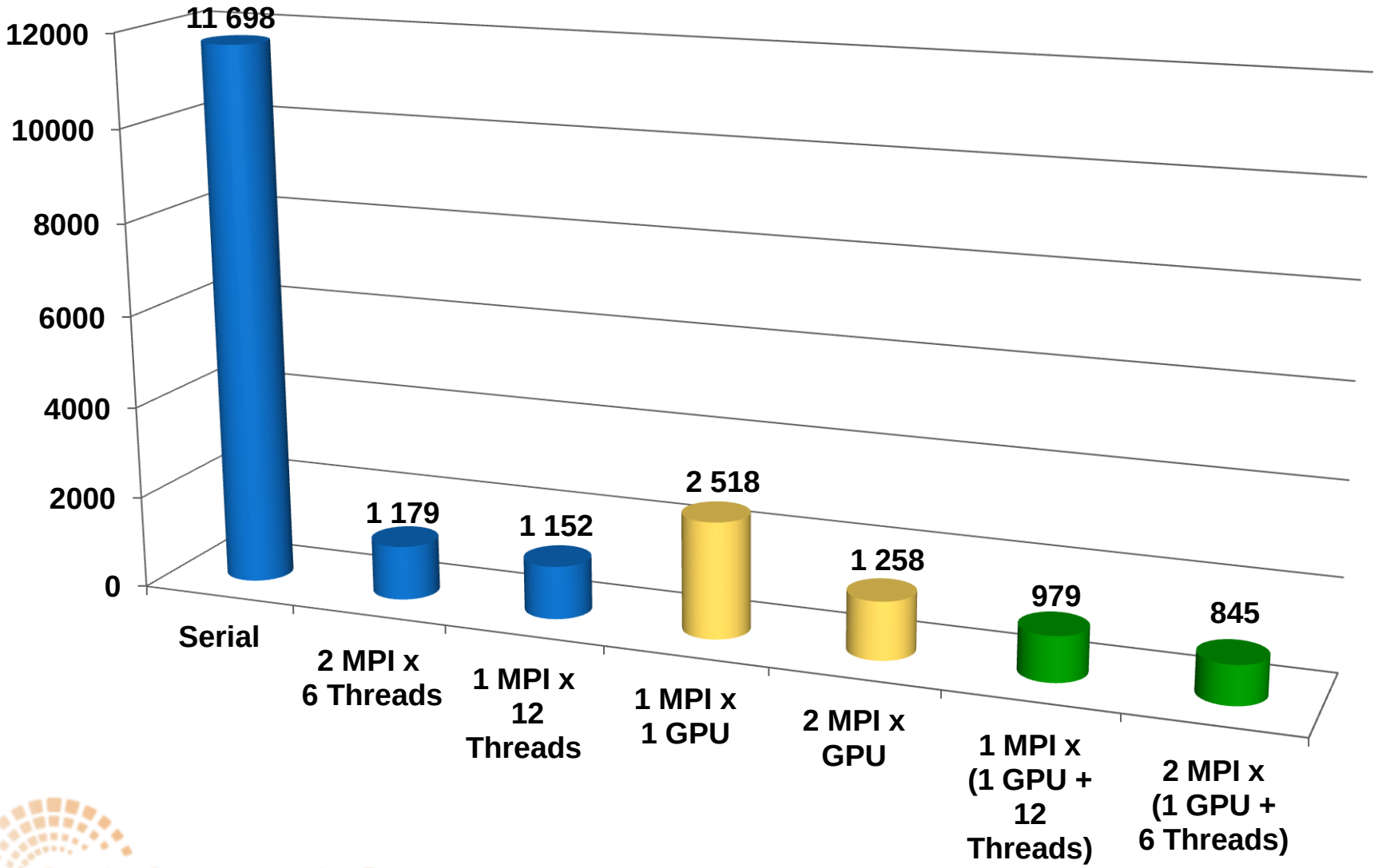
© ИПМ ИМ. М.В. КЕЛДЫША РАН

Ускорение выполнения 100 итераций программы Композит на сетке 261x261x6



© ИПМ ИМ. М.В. КЕЛДЫША РАН

Время работы программы Композит на сетке 261x261x6 на узле K-100 (2 x Intel Xeon X5670 + 2 x GPU NVIDIA Fermi C2050)



© ИПМ ИМ. М.В. КЕЛДЫША РАН

Распараллеливание прикладных задач, использующих неструктурированные сетки

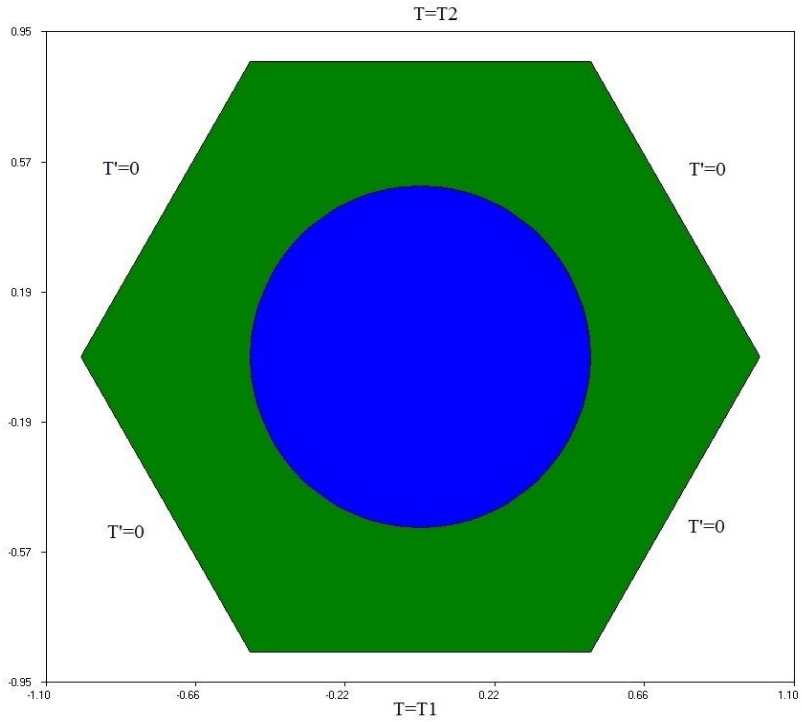
В 2016 году сформулированы предложения по расширению DVMH-модели:

- 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

© ИПМ ИМ. М.В. КЕЛДЫША РАН

Двумерная задача теплопроводности в шестиграннике

Область состоит из двух материалов с различными коэффициентами температуропроводности.

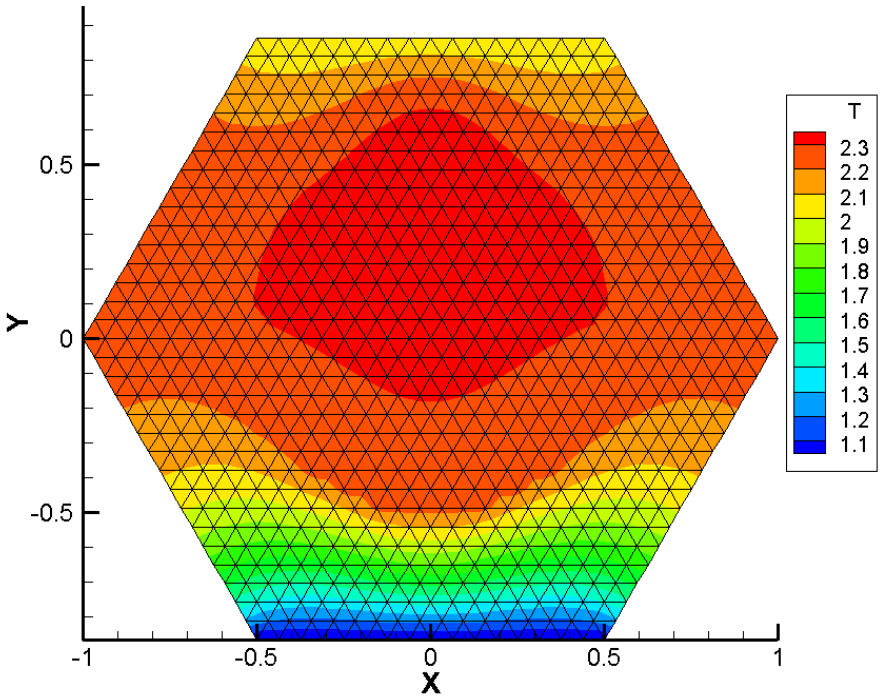



```
do i = 1, np2
  nn = ii(i)
  nb = npa(i)
  if (nb.ge.0) then
    s1 = FS(xp2(i),yp2(i),tv)
    s2 = 0d0
    do j = 1, nn
      j1 = jj(j,i)
      s2 = s2 + aa(j,i) * tt1(j1)
    enddo
    s0 = s1 + s2
    tt2(i) = tt1(i) + tau * s0
  else if (nb.eq.-1) then
    tt2(i) = vtemp1
  else if (nb.eq.-2)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


© ИПМ ИМ. М.В. КЕЛДЫША РАН

Двумерная задача теплопроводности в шестиграннике

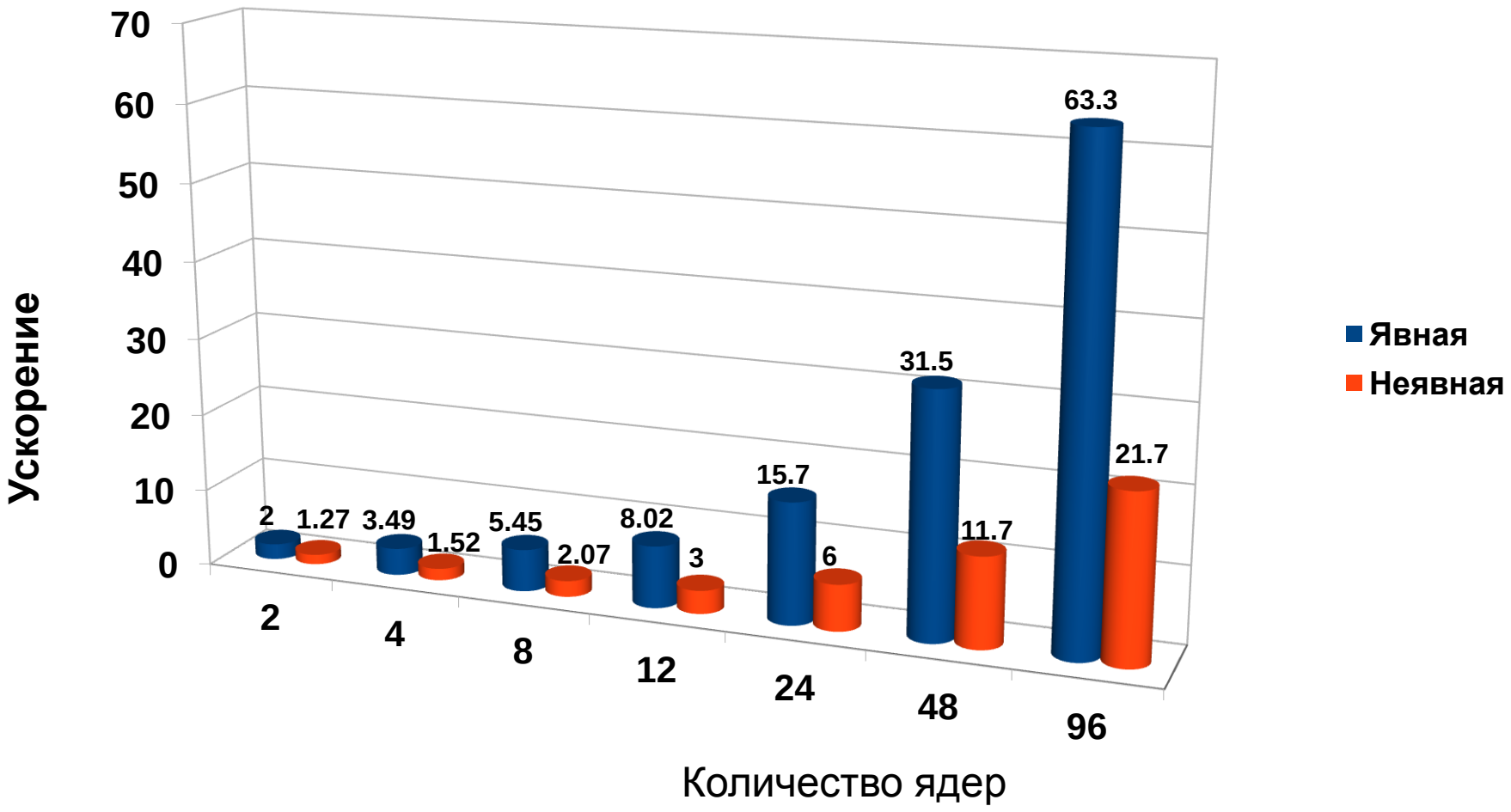
Стационарное распределение температуры




```
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
```

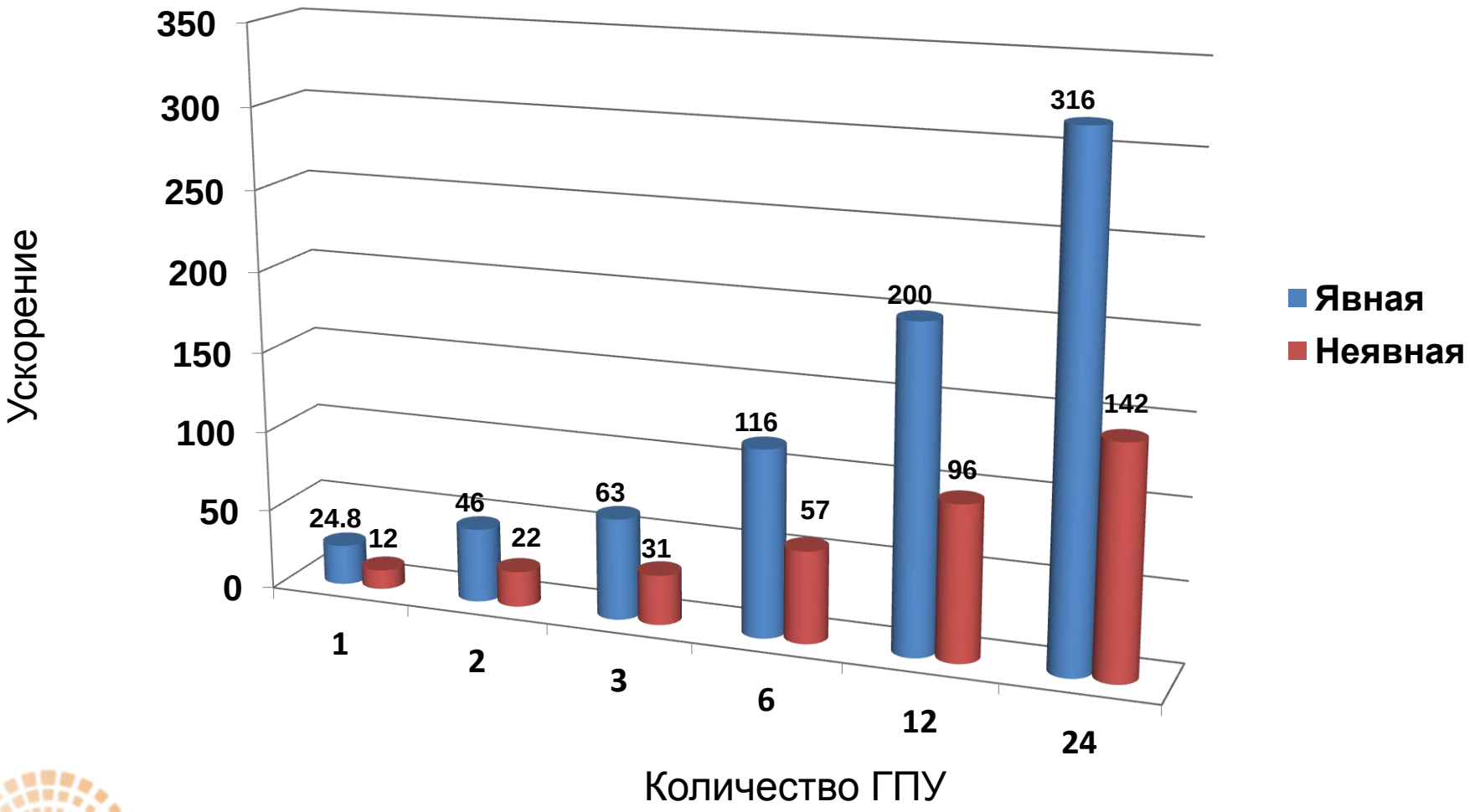

© ИПМ ИМ. М.В. КЕЛДЫША РАН

Ускорение на ЦПУ для сетки 8 млн. узлов



© ИПМ ИМ. М.В. КЕЛДЫША РАН

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



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;
}
```


© ИПМ ИМ. М.В. КЕЛДЫША РАН

Опыт использования средств DVMH в MPI-программах

- HyperbolicSolver2D. Солвер явной схемы систем гиперболических уравнений - часть большого развитого комплекса вычислительных программ (В.А. Гасилов, А.С. Болдарев)
  - C++, 39000 LOC, шаблоны, полиморфизм, и т.п.
  - богатая платформа базовых понятий и структур данных
- 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

Выводы

- DVMH-модель параллельного программирования рассчитана на работу со структурированными и неструктурированными сетками, позволяет осуществлять дополнительное распараллеливание MPI-программ
- Использование DVM-системы и системы САПФОР существенно упрощает процесс разработки параллельных программ для гетерогенных вычислительных кластеров

http://dvm-system.org

Вопросы, замечания?

http://dvm-system.org