Особенности реализации с помощью DVM-системы программ, использующих нерегулярные сетки

В.А. Бахтин, А.С. Колганов, В.А. Крюков, Н.В.Поддерюгина, С.В. Поляков, **М.Н. Притула**

pritmick@yandex.ru

Институт прикладной математики им. М.В. Келдыша РАН

План доклада

- DVMH-модель параллельного программирования
- Применение имеющихся средств для задач на нерегулярных сетках
- Предложение по расширению модели DVMH для полноценной работы с нерегулярными сетками

```
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", "wb");
  #pragma dvm get_actual(B)
  fwrite(B, sizeof(double), L * L, f);
  fclose(f);
  return 0;
}
```

Алгоритм Якоби в модели DVMH

Средства программирования

C-DVMH = Язык Си + 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

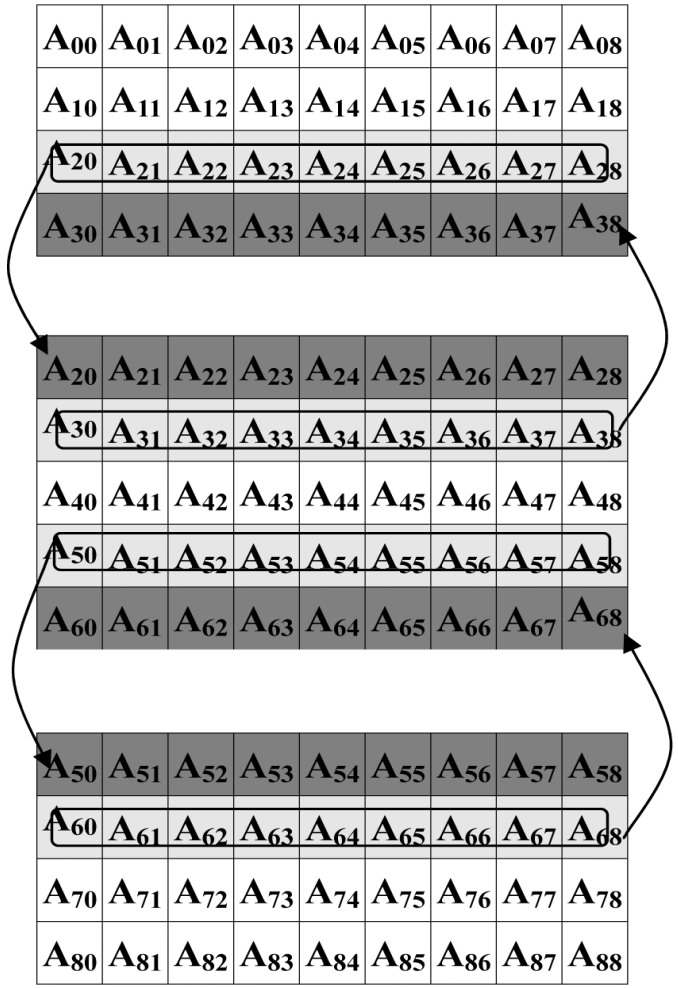
Состав DVM-системы

Система состоит из следующих компонент:

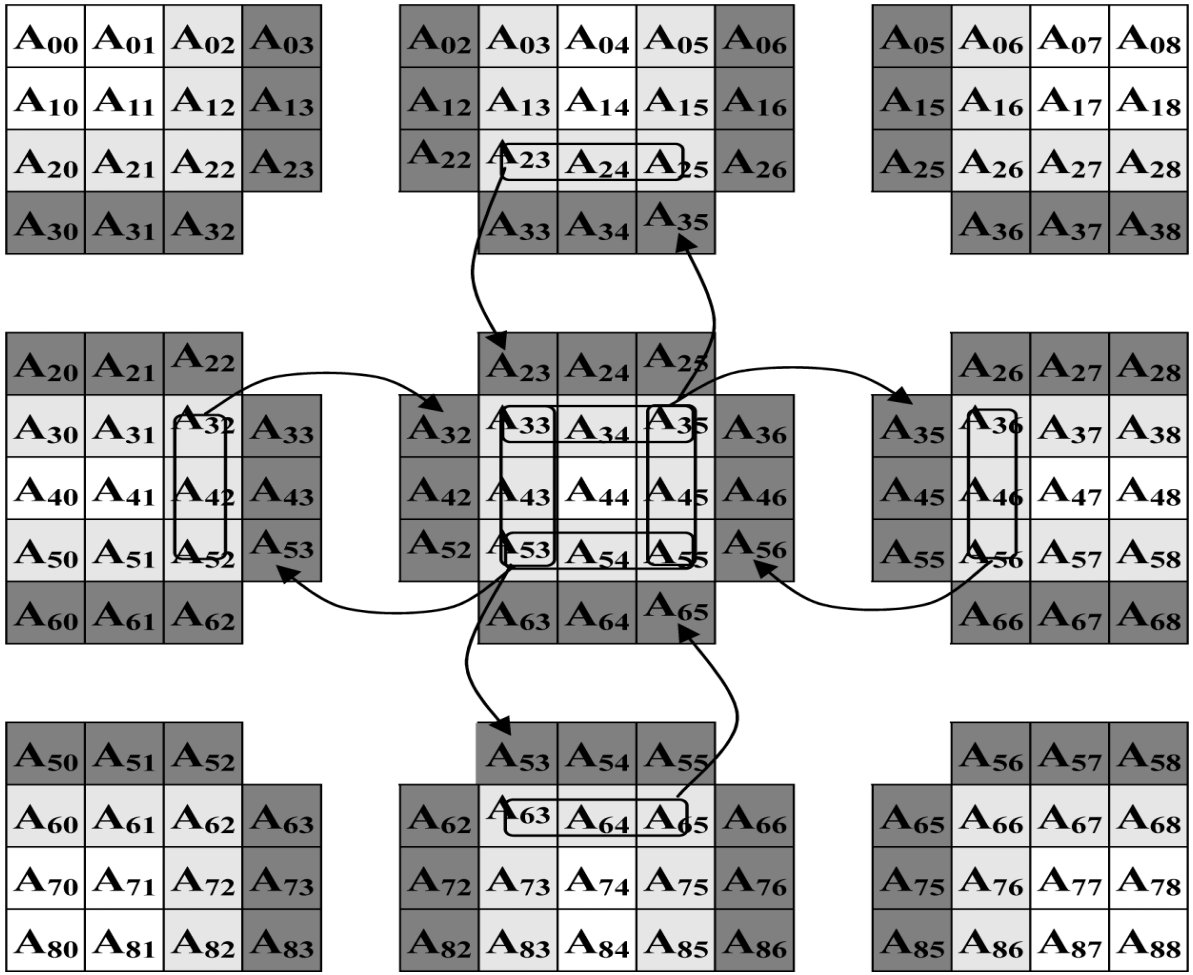
- Компилятор Fortran-DVMH
- Компилятор C-DVMH
- Библиотека поддержки LibDVMH
- DVMH-отладчик
- Анализатор производительности DVMH-программ

Распределение массива

./dvm run 3 1 jac



./dvm run 3 3 jac

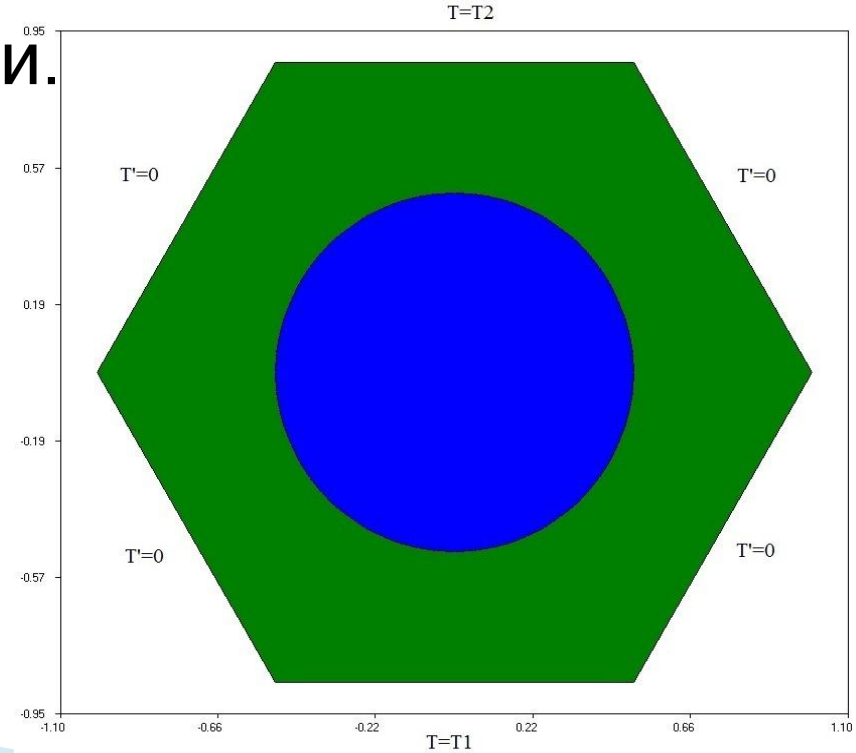


Нерегулярные сетки

- Варьирование подробности сетки
- Работа со сложной геометрией области
- Обобщение вычислительных кодов
- Более сложная для анализа и распараллеливания структура программ:
  - Косвенная индексация
  - Переменное число “соседей”

Рассматриваемая задача

- Двумерная задача теплопроводности с постоянным, но разрывным коэффициентом в шестиграннике.
- Область состоит из двух материалов с различными коэффициентами температуропроводности.



Рассматриваемая задача

- Массивы величин одномерные - `tt1,tt2`
- Переменное число “соседей” - `ii`
- Связи задаются массивом - `jj`

```
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
```

Ограничение для возможности распараллеливания в DVMH

- Существует такое небольшое M, при котором все соседи ячейки с номером n имеют номера, отличающиеся от n не более, чем на M
- Предлагается считать допустимым такое M, при котором M/N стремится к нулю при измельчении сетки (N – общее количество ячеек в сетке)

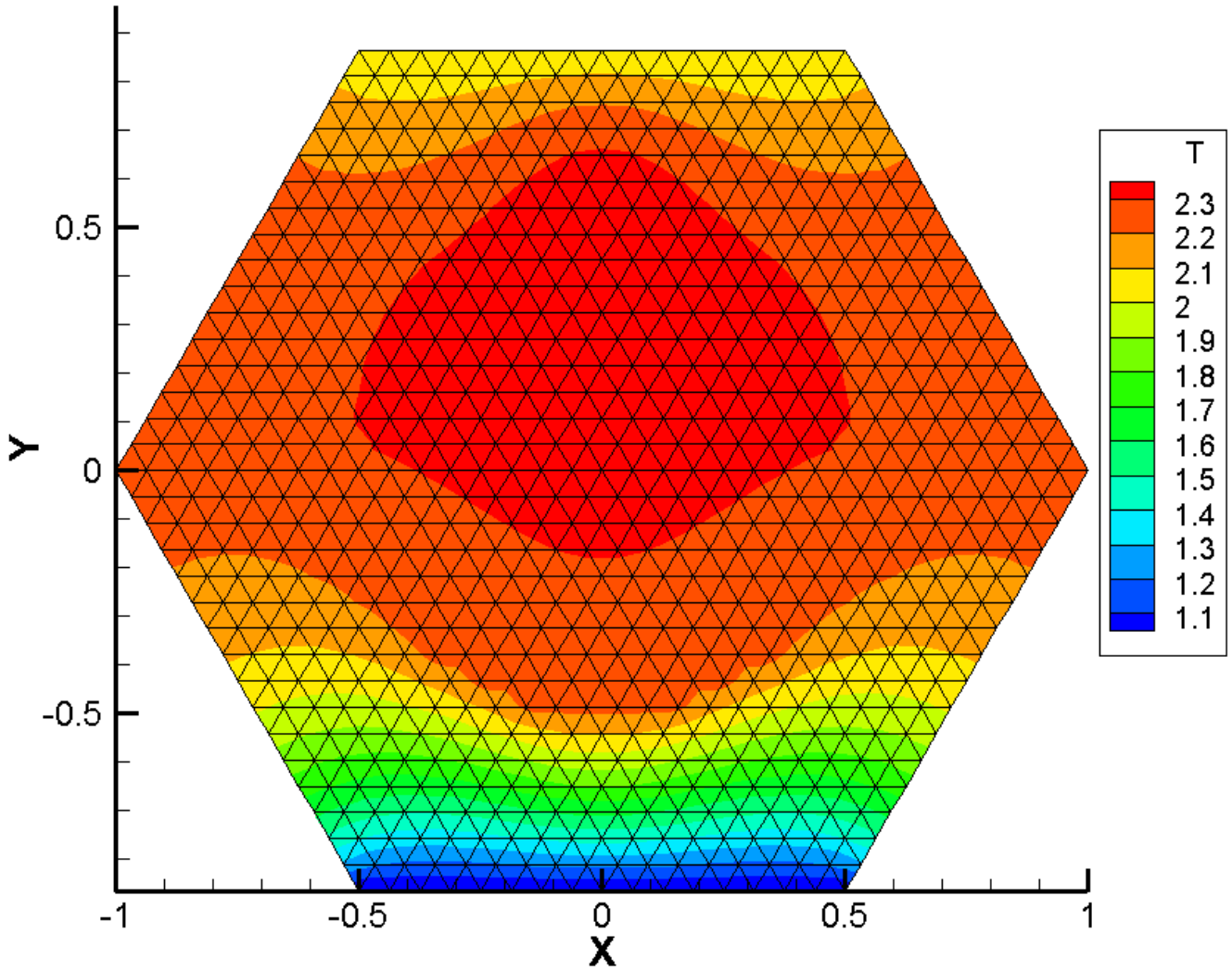
Распараллеливание в DVMH

- Все расчетные и индексные массивы были распределены
- Добавлен в программу **расчет ширины теневых граней для конкретного запуска**
  - Значение одинаково на всех процессорах
  - Только непрерывные участки, непосредственно примыкающие к локальной части
- Добавлены спецификации обменов и распределения вычислений

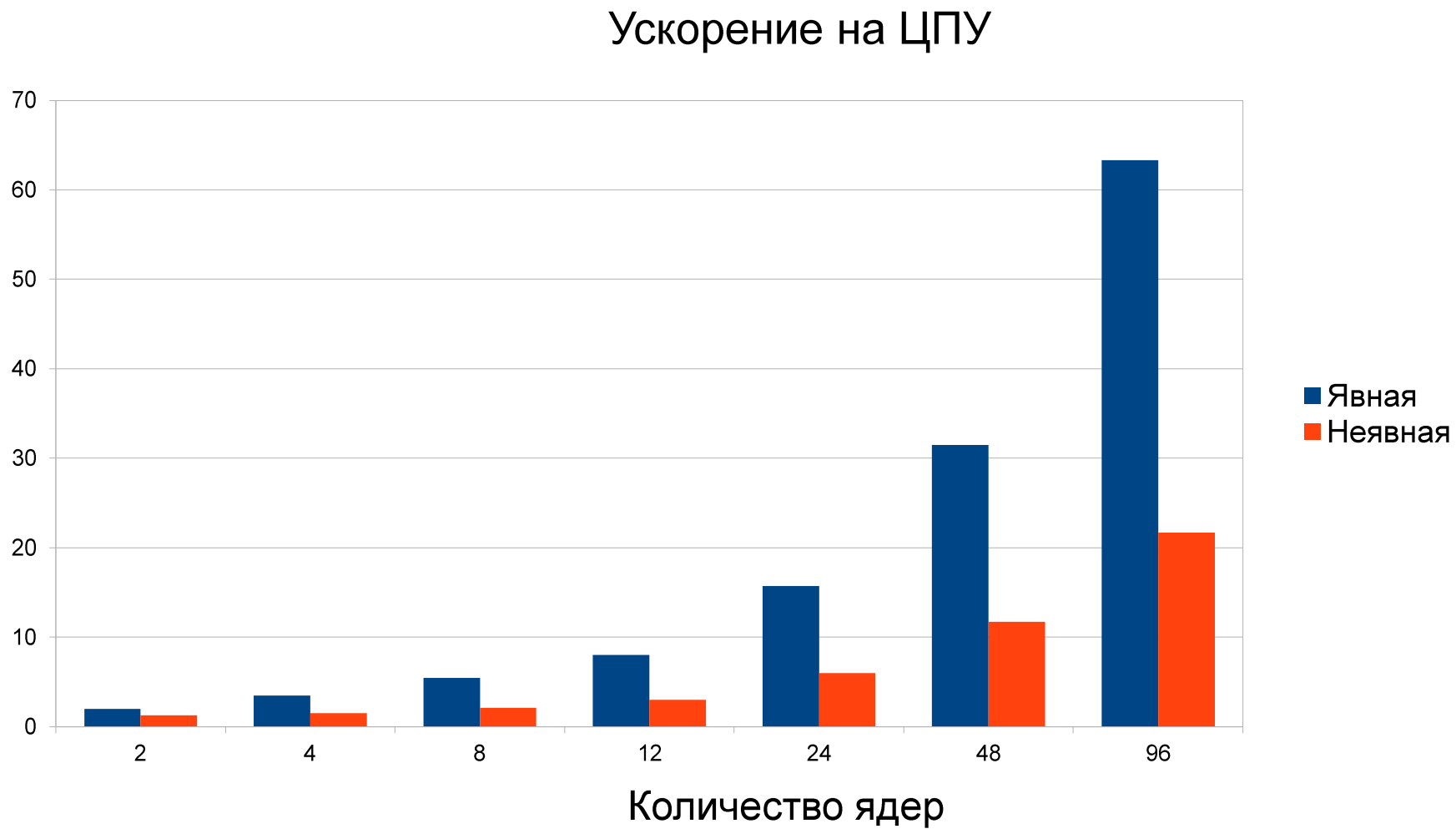
Расчет ширины теневых граней

```
      nb1 = 1
      nb2 = np2
! Trying to figure out the size of the shadow edges
      shadL = 0
      shadR = 0
!DVM$ PARALLEL(i) ON npa_d(i), NEW(nb1, nb2)
      do i = nb1, nb2
      enddo
!DVM$ PARALLEL(i)ON npa_d(i),reduction(max(shadL),max(shadR))
         do i = 1, np2
           nn = ii(i)
           do j = 1, nn
             j1 = jj(j,i)
             if (j1.lt.nb1) then
               shadL = max(shadL, nb1 - j1)
             elseif (j1.gt.nb2) then
               shadR = max(shadR, j1 - nb2)
             endif
                  enddo
                           enddo
```

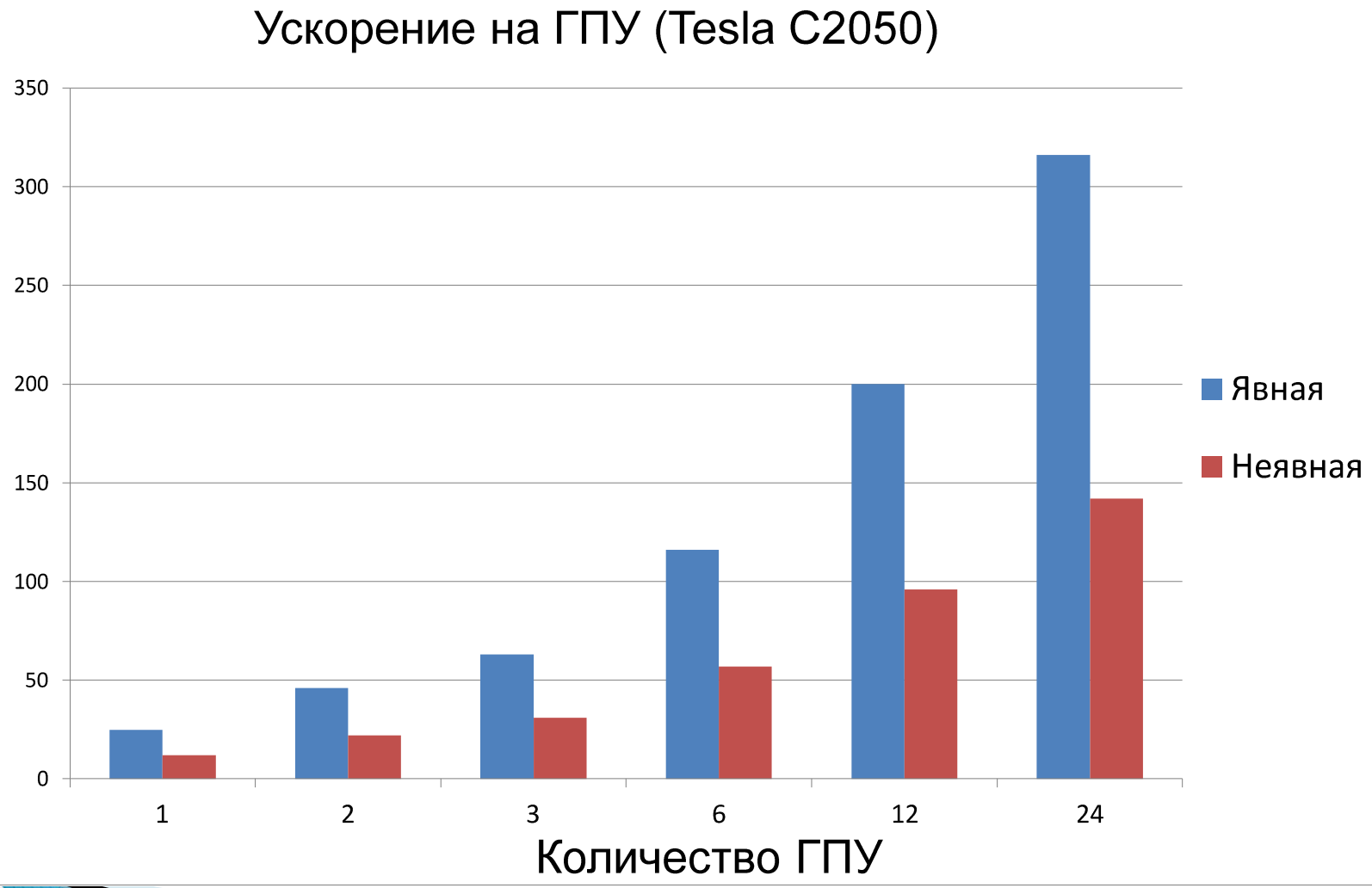
Расчетная сетка



Полученные ускорения, 8 млн узлов



Полученные ускорения, 8 млн узлов



Обнаруженные недостатки модели DVMH

- Использование механизма теневых граней приводит к избыточности обменов и потребления памяти
- Необходимость специального упорядочения (нумерации) сеточных элементов
- Распределение только блочное (и одномерное вследствие одномерности массива величин)
- Невозможность согласованного распределения сеточных элементов (ячейки, узлы, ребра)

Обзор новых возможностей

- 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[neigh[i][0]:neigh[i][numneigh[i]-1]] with nodes[@i]] name=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 простых циклов нет необходимости в ресурсоёмкой подготовке:
  - Переменная цикла используется только для индексации связанных измерений
  - Индексация поэлементно-распределенных измерений только переменной цикла
- `#pragma dvm parallel([i] on A[i])` **`byref`**

```
for (int i = 0; i < N; i++)
    A[i] = 0;
```

Переход к локальной индексации

- Для быстрых косвенных обращений вводится *область косвенной индексации*:
  - Говорит о том, что следует подготовить косвенную индексацию по локальным индексам для указанных индексных массивов
  - В циклах, проводимых по локальным индексам:
    - Разрешается индексировать указанными в регионе индексными массивами (а не только переменной цикла)
    - Запрещается использовать значения указанных индексных массивов в иных целях
- `reference(tt1[:],tt2[:] <= jj)`

Выводы

- Созданы экспериментальные параллельные DVM-версии программ для решения краевой задачи для двумерного квазилинейного параболического уравнения, записанного в дивергентной форме в различной постановке на неструктурированных треугольных сетках.
- Параллельные версии продемонстрировали следующие ускорения (8 млн узлов сетки). Явная схема: на 12-ядерном узле - 8 раз, на 1 ГПУ 25 раз, на 24 ГПУ - 316 раз. Неявная схема: на 12-ядерном узле - 3 раза, на 1 ГПУ 22 раза, на 24 ГПУ — 142 раза.
- Предложено расширение модели DVMH, реализация которого позволит существенно расширить класс программ, которые могут быть разработаны на языках Fortran-DVMH и C-DVMH и эффективно выполняться на суперкомпьютрах различной архитектуры.

Вопросы, замечания?

СПАСИБО !

http://dvm-system.org
dvm@keldysh.ru