

Расширение возможностей DVM-системы для решения задач, использующих нерегулярные сетки

В.А. Бахтин, А.С. Колганов, В.А. Крюков, Н.В.Поддержюгина, М.Н. Притула

Институт прикладной математики им. М.В. Келдыша РАН

<http://dvm-system.org>



План доклада

- ▶ Программы на регулярных и нерегулярных сетках;
- ▶ DVMH-модель параллельного программирования;
- ▶ Использование средств DVMH в MPI-программах;
- ▶ Новые возможности работы с нерегулярными сетками в компиляторе C-DVMH.

Программы на регулярных сетках

Проблемы

- Единый шаг сетки по расчетной области – нет гибкости, неподъемные требования по памяти и вычислительной мощности при измельчении;
- Реализация численных методов зачастую привязана к форме сетки – двумерные, трехмерные, декартовы, цилиндрические, и т.д. Тем самым сложно заменять геометрию.

Положительные стороны

- Отношение соседства и пространственные координаты не хранятся явно – экономия памяти;
- Обращения к массивам с константными смещениями – свобода для компиляторных оптимизаций, ясность для распараллеливания (в том числе автоматического).

Программы на нерегулярных сетках

Положительные стороны

- Свобода вариации подробности сетки – поддержание нужной степени измельчения в отдельных частях области;
- Хорошие возможности по переиспользованию вычислительного кода, свобода в выборе формы расчетных областей.

Проблемы

- Отношение соседства и пространственные координаты приходится хранить явно;
- Косвенная индексация при обращениях к массивам – препятствие для компиляторных оптимизаций, сложность для распараллеливания (в особенности автоматического).



```

double A[L][L];

double B[L][L];

int main(int argc, char *argv[]) {
    for(int it = 0; it < ITMAX; it++) {

        {

            for (int i = 1; i < L - 1; i++)
                for (int j = 1; j < L-1; j++)
                    A[i][j] = B[i][j];

            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.;
        }
    }
    FILE *f = fopen("jacobi.dat", "wb");

    fwrite(B, sizeof(double), L * L, f);
    fclose(f);
    return 0;
}

```

Алгоритм Якоби

```
#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++) {
```

```
        {
```

```
            for (int i = 1; i < L - 1; i++)  
                for (int j = 1; j < L - 1; j++)  
                    A[i][j] = B[i][j];
```

```
            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.;
```

```
        }
```

```
    }
```

```
    FILE *f = fopen("jacobi.dat", "wb");
```

```
    fwrite(B, sizeof(double), L * L, f);
```

```
    fclose(f);
```

```
    return 0;
```

```
}
```

**Алгоритм Якоби
в модели DVMH**

```
#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 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.;
```

```
    }
```

```
}
```

```
FILE *f = fopen("jacobi.dat", "wb");
```

```
fwrite(B, sizeof(double), L * L, f);
```

```
fclose(f);
```

```
return 0;
```

```
}
```

**Алгоритм Якоби
в модели DVMH**

```
#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.;
```

```
        }
```

```
    }
```

```
    FILE *f = fopen("jacobi.dat", "wb");
```

```
    fwrite(B, sizeof(double), L * L, f);
```

```
    fclose(f);
```

```
    return 0;
```

```
}
```

**Алгоритм Якоби
в модели DVMH**


```
#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.;
```

```
        }
```

```
    }  
    FILE *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 + спец. прагмы

Fortran-DVMH = Язык Фортран 95 + спец. комментарии

- ▶ Специальные комментарии и прагмы являются высокоуровневыми спецификациями параллелизма в терминах последовательной программы;
- ▶ Отсутствуют низкоуровневые передачи данных и синхронизации в коде программы;
- ▶ 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 (директивы **distribute / align**);
- ▶ Распределение витков цикла между вычислительными устройствами (директива **parallel**);
- ▶ Спецификация параллельно выполняющихся секций программы (параллельных задач) и отображение их на процессоры (директива **task**);
- ▶ Организация эффективного доступа к удаленным данным, расположенным на других вычислительных устройствах (спецификации **shadow / across / remote**).

Спецификации параллельного выполнения программы

- ▶ Организация эффективного выполнения редукционных операций – глобальных операций с расположенными на различных вычислителях данными (спецификация `reduction: max/min/sum/maxloc/minloc/...`);
- ▶ Определение фрагментов программы (регионов) для возможного выполнения на ускорителях и многоядерных CPU (директива `region`);
- ▶ Управление перемещением данных между памятью CPU и памятью GPU (директивы `actual` / `get_actual`).

Компоненты DVM-системы

- ▶ Компилятор Fortran-DVMH;
- ▶ Компилятор C-DVMH;
- ▶ Библиотека поддержки Lib-DVMH
(DVMH Run Time System);
- ▶ Отладчик DVMH-программ;
- ▶ Анализатор производительности DVMH-программ.



Динамическое распределение массива

./dvm run 3 1 jac

A ₀₀	A ₀₁	A ₀₂	A ₀₃	A ₀₄	A ₀₅	A ₀₆	A ₀₇	A ₀₈
A ₁₀	A ₁₁	A ₁₂	A ₁₃	A ₁₄	A ₁₅	A ₁₆	A ₁₇	A ₁₈
A ₂₀	A ₂₁	A ₂₂	A ₂₃	A ₂₄	A ₂₅	A ₂₆	A ₂₇	A ₂₈
A ₃₀	A ₃₁	A ₃₂	A ₃₃	A ₃₄	A ₃₅	A ₃₆	A ₃₇	A ₃₈

A ₂₀	A ₂₁	A ₂₂	A ₂₃	A ₂₄	A ₂₅	A ₂₆	A ₂₇	A ₂₈
A ₃₀	A ₃₁	A ₃₂	A ₃₃	A ₃₄	A ₃₅	A ₃₆	A ₃₇	A ₃₈
A ₄₀	A ₄₁	A ₄₂	A ₄₃	A ₄₄	A ₄₅	A ₄₆	A ₄₇	A ₄₈
A ₅₀	A ₅₁	A ₅₂	A ₅₃	A ₅₄	A ₅₅	A ₅₆	A ₅₇	A ₅₈
A ₆₀	A ₆₁	A ₆₂	A ₆₃	A ₆₄	A ₆₅	A ₆₆	A ₆₇	A ₆₈

A ₅₀	A ₅₁	A ₅₂	A ₅₃	A ₅₄	A ₅₅	A ₅₆	A ₅₇	A ₅₈
A ₆₀	A ₆₁	A ₆₂	A ₆₃	A ₆₄	A ₆₅	A ₆₆	A ₆₇	A ₆₈
A ₇₀	A ₇₁	A ₇₂	A ₇₃	A ₇₄	A ₇₅	A ₇₆	A ₇₇	A ₇₈
A ₈₀	A ₈₁	A ₈₂	A ₈₃	A ₈₄	A ₈₅	A ₈₆	A ₈₇	A ₈₈

./dvm run 3 3 jac

A ₀₀	A ₀₁	A ₀₂	A ₀₃
A ₁₀	A ₁₁	A ₁₂	A ₁₃
A ₂₀	A ₂₁	A ₂₂	A ₂₃
A ₃₀	A ₃₁	A ₃₂	

A ₀₂	A ₀₃	A ₀₄	A ₀₅	A ₀₆
A ₁₂	A ₁₃	A ₁₄	A ₁₅	A ₁₆
A ₂₂	A ₂₃	A ₂₄	A ₂₅	A ₂₆
A ₃₃	A ₃₄	A ₃₅		

A ₀₅	A ₀₆	A ₀₇	A ₀₈
A ₁₅	A ₁₆	A ₁₇	A ₁₈
A ₂₅	A ₂₆	A ₂₇	A ₂₈
A ₃₆	A ₃₇	A ₃₈	

A ₂₀	A ₂₁	A ₂₂	
A ₃₀	A ₃₁	A ₃₂	A ₃₃
A ₄₀	A ₄₁	A ₄₂	A ₄₃
A ₅₀	A ₅₁	A ₅₂	A ₅₃
A ₆₀	A ₆₁	A ₆₂	

A ₂₃	A ₂₄	A ₂₅	
A ₃₂	A ₃₃	A ₃₄	A ₃₅
A ₄₂	A ₄₃	A ₄₄	A ₄₅
A ₅₂	A ₅₃	A ₅₄	A ₅₅
A ₆₃	A ₆₄	A ₆₅	

A ₂₆	A ₂₇	A ₂₈	
A ₃₅	A ₃₆	A ₃₇	A ₃₈
A ₄₅	A ₄₆	A ₄₇	A ₄₈
A ₅₅	A ₅₆	A ₅₇	A ₅₈
A ₆₆	A ₆₇	A ₆₈	

A ₅₀	A ₅₁	A ₅₂	
A ₆₀	A ₆₁	A ₆₂	A ₆₃
A ₇₀	A ₇₁	A ₇₂	A ₇₃
A ₈₀	A ₈₁	A ₈₂	A ₈₃

A ₅₃	A ₅₄	A ₅₅	
A ₆₂	A ₆₃	A ₆₄	A ₆₅
A ₇₂	A ₇₃	A ₇₄	A ₇₅
A ₈₂	A ₈₃	A ₈₄	A ₈₅

A ₅₆	A ₅₇	A ₅₈	
A ₆₅	A ₆₆	A ₆₇	A ₆₈
A ₇₅	A ₇₆	A ₇₇	A ₇₈
A ₈₅	A ₈₆	A ₈₇	A ₈₈

Использование средств DVMH в MPI-программе: причины

- ▶ Уже накоплен фонд и опыт написания параллельных программ для кластера;
- ▶ Модель DVMH предполагает распараллеливание последовательных программ;
- ▶ Пользователь не желает отказываться от своей параллельной программы;
- ▶ Программа распараллелена вручную так, как не удастся (или потребует трюков) сделать средствами DVMH (например – она на нерегулярных сетках);
- ▶ Есть желание добавить параллелизм на нитях CPU и/или на GPU, а также воспользоваться инструментарием DVMH (отладка и профилирование).



Использование средств DVMH в MPI-программе: результаты

- ▶ Добавлен новый режим работы DVM-системы локально в каждом процессе;
- ▶ Добавлена конструкция нераспределенного параллельного цикла;
- ▶ Инкрементальное распараллеливание и быстрая оценка эффективности DVMH-модели на нитях CPU и GPU перед проведением полного распараллеливания;
- ▶ Возможность использовать DVMH-распараллеливание внутри узла кластера в готовых MPI-программах.



Использование средств DVMH в MPI-программе: опыт

- ▶ Солвер явной схемы – часть большого развитого комплекса вычислительных программ:
 - C++, 39 000 LOC, шаблоны, полиморфизм, и т.п.;
 - богатая платформа базовых понятий и структур данных;
- ▶ Модификации только локальные, в одном модуле в ~3000 строк, которые сводятся к добавлению нескольких директив (около 10);
- ▶ Получено ускорение
 - 2 CPU Intel Xeon X5670 по 6 ядер – 9.8 раз;
 - GPU NVidia GTX Titan (Kepler) – 18 раз.



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;
- ▶ Сохранение быстрого доступа к распределенным массивам с помощью механизмов перехода на локальную индексацию;
- ▶ Возможность реорганизации данных – оптимизации шаблона доступа к памяти путем изменения порядка хранения локальных элементов.

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)`

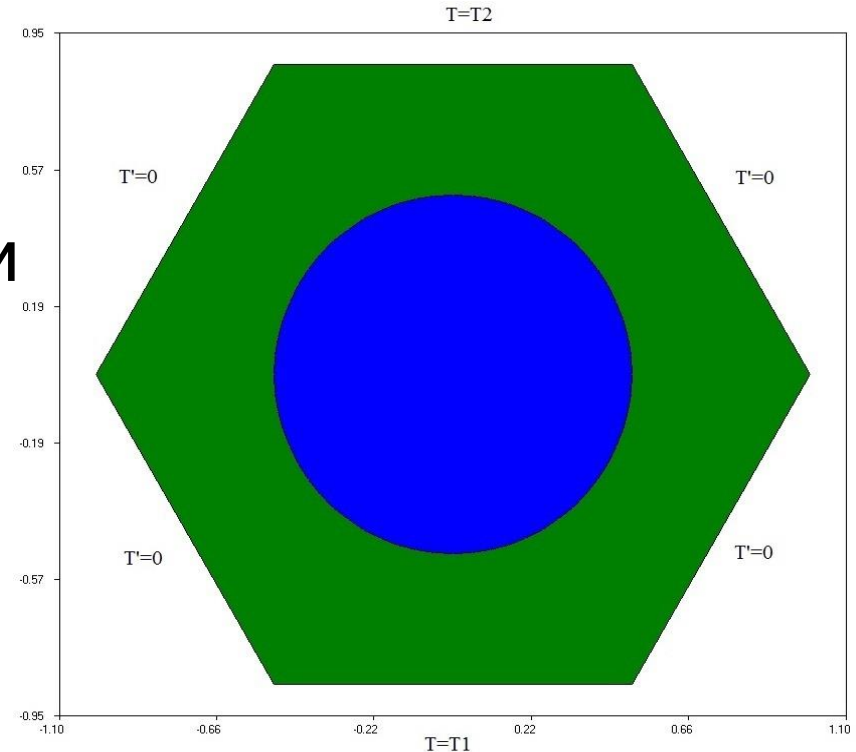


Переход к локальной индексации

- ▶ Процедура для перевода глобального (исходного) индекса в локальный (непосредственно для доступа к памяти) слишком долгая;
- ▶ Для блочных распределений глобальный и локальный индексы 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.

Рассматриваемая задача

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



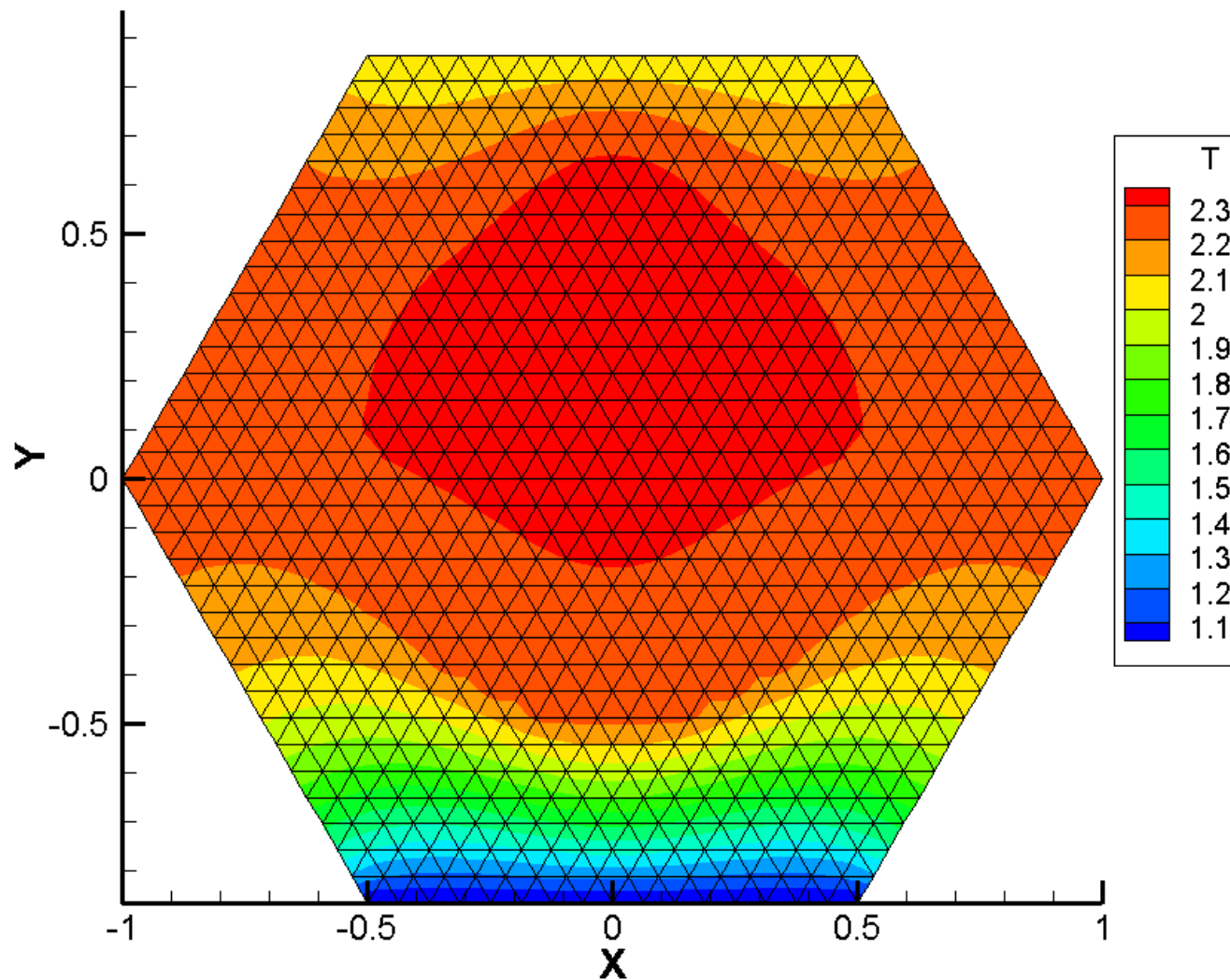
Рассматриваемая задача

- ▶ 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
```



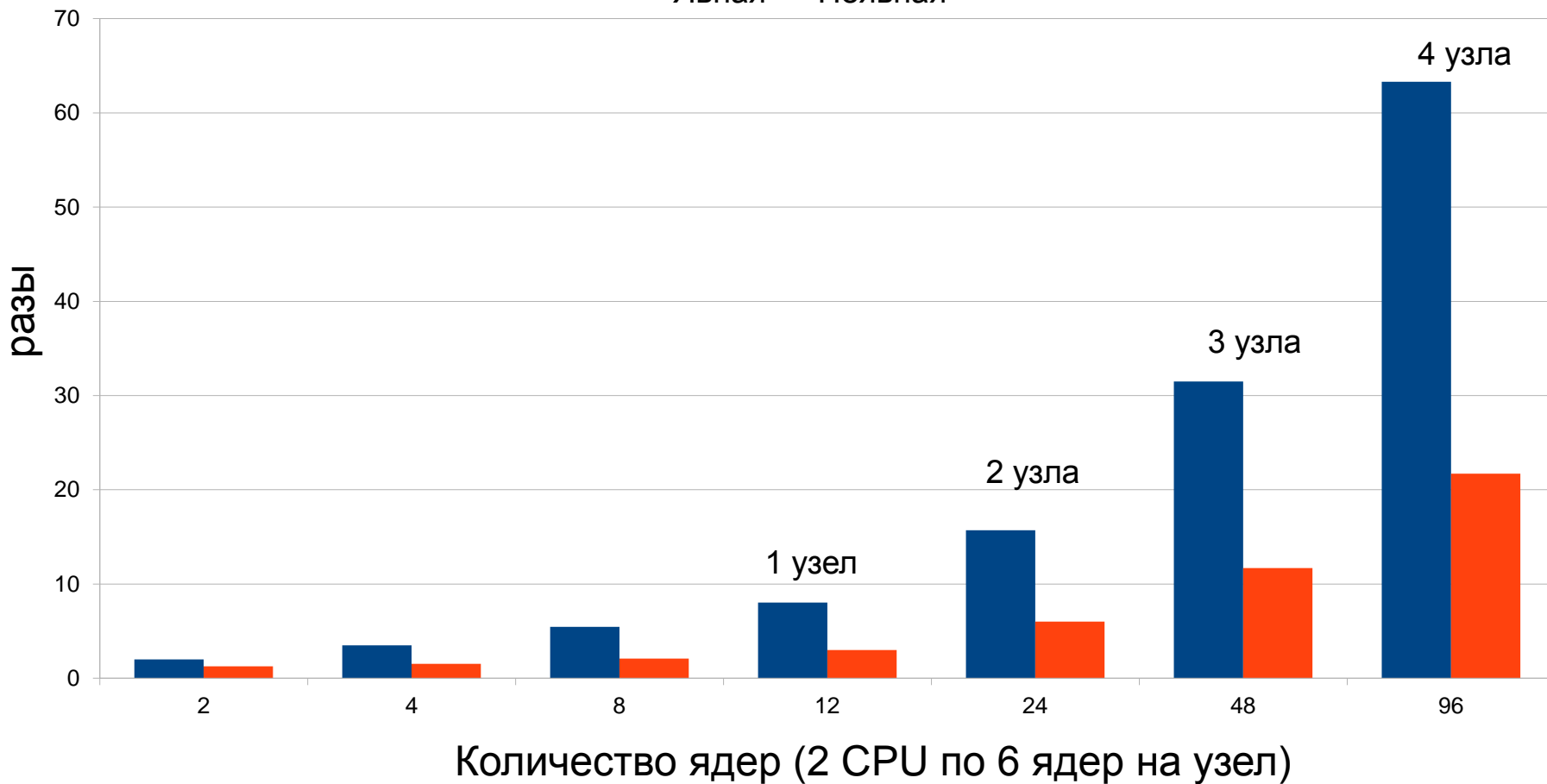
Расчетная сетка



Полученные ускорения, 8 млн узлов

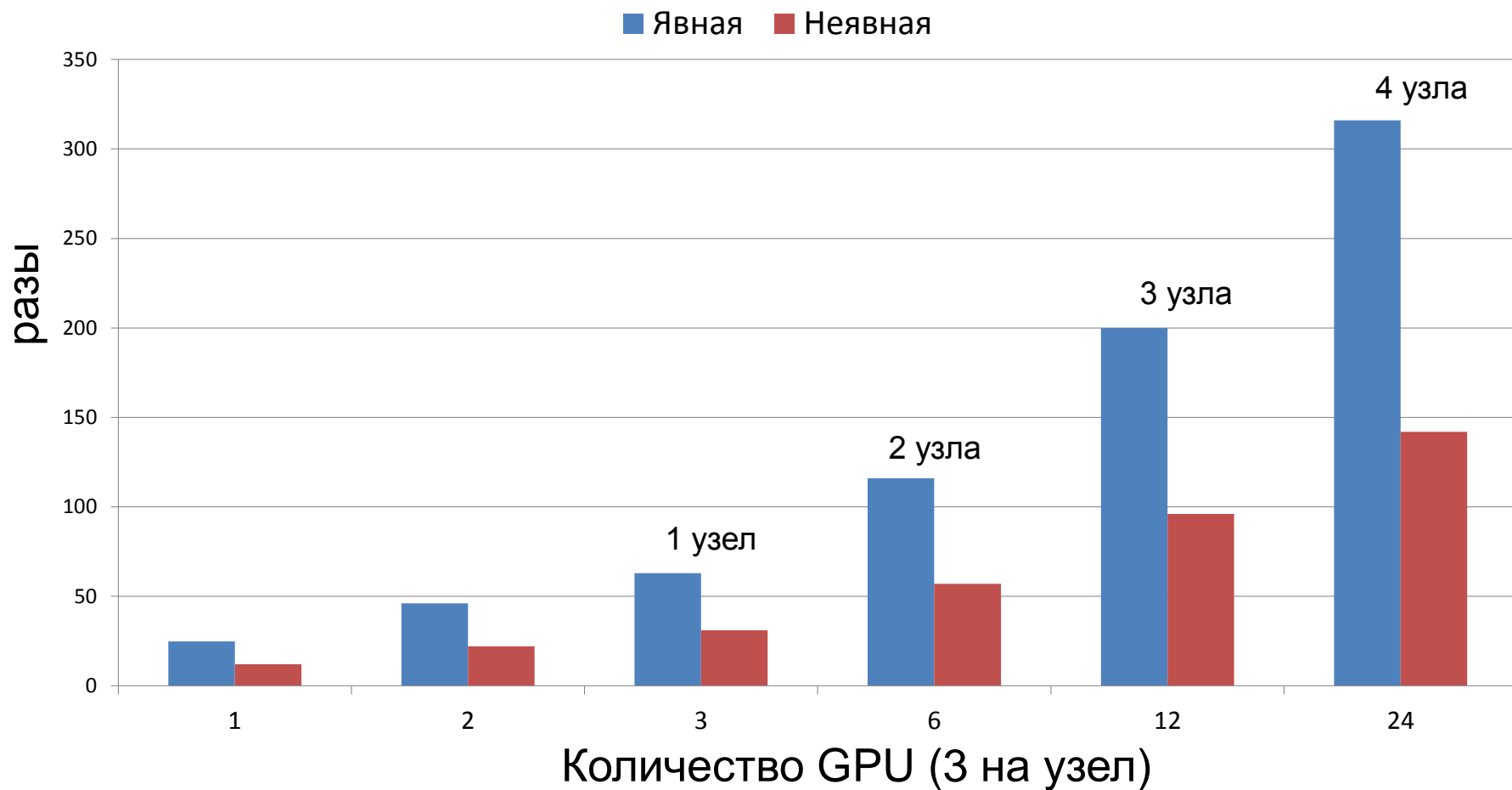
Ускорение на CPU Intel Xeon X5670

■ Явная ■ Неявная



Полученные ускорения, 8 млн узлов

Ускорение на GPU Nvidia Tesla C2050



Выводы

- ▶ Сделаны шаги по расширению возможностей DVM-системы для решения задач, использующих нерегулярные сетки и позволяющие распараллелить их на кластер с многоядерными и графическими процессорами:
 - возможность перехода на ручное распределение данных (например, средствами MPI);
 - новые экспериментальные конструкции языка C-DVMH и их реализация в компиляторе и системе поддержки;
- ▶ В дальнейшем предполагается существенно расширить новые средства и улучшить их интеграцию с DVMH-программами, использующие блочно-распределенные данные.

Вопросы, замечания?

СПАСИБО !

сайт: <http://dvm-system.org>

почта: dvm@keldysh.ru