МЕЖДУНАРОДНАЯ НАУЧНАЯ КОНФЕРЕНЦИЯ

ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ ТЕХНОЛОГИИ 2017

Автоматизированное распараллеливание задачи моделирования распространения упругих волн в средах со сложной 3D геометрией поверхности на кластеры разной архитектуры

4 апреля 2017г. | Казань

Н.А. Катаев, А.С. Колганов, П.А. Титов

Институт прикладной математики им. М.В. Келдыша РАН

Факультет вычислительной математики и кибернетики МГУ им. М.В. Ломоносова

Институт вычислительной математики и математической геофизики СО РАН

Автоматизация распараллеливания

- Распараллеливание программ – процесс их адаптации для эффективного исполнения на вычислительной системе параллельной архитектуры. Как правило заключается либо в **переписывании программ** на специальный язык, либо во **вставке специальной разметки**.

- Автоматическое распараллеливание – оптимизация программы компилятором, состоящая в автоматическом её преобразовании для эффективного выполнения на параллельном компьютере, например, на SMP или NUMA машине.

- Автоматизация распараллеливания – процесс оптимизации программы компилятором, состоящий в **автоматизированном ее отображении** в параллельную программу, в котором **пользователь принимает активное участие**.

Средства параллельного программирования

DVM-система

- Создана в Институте прикладной математики им. М.В. Келдыша РАН

- Аббревиатура DVM:
  Distributed Virtual Memory
  Distributed Virtual Machine

- Существует для двух языков: *C-DVMH* и *Fortran-DVMH*

- Предназначена для использования на кластерах с аппаратурой различной архитектуры (GPU NVidia, Intel Xeon Phi, multicore CPUs).

Компоненты DVM-системы

- Компилятор Fortran-DVMH
- Компилятор C-DVMH
- Библиотека поддержки Lib-DVMH:

  DVMH Run Time System
- Отладчик DVMH-программ
- Анализатор производительности DVMH-программ

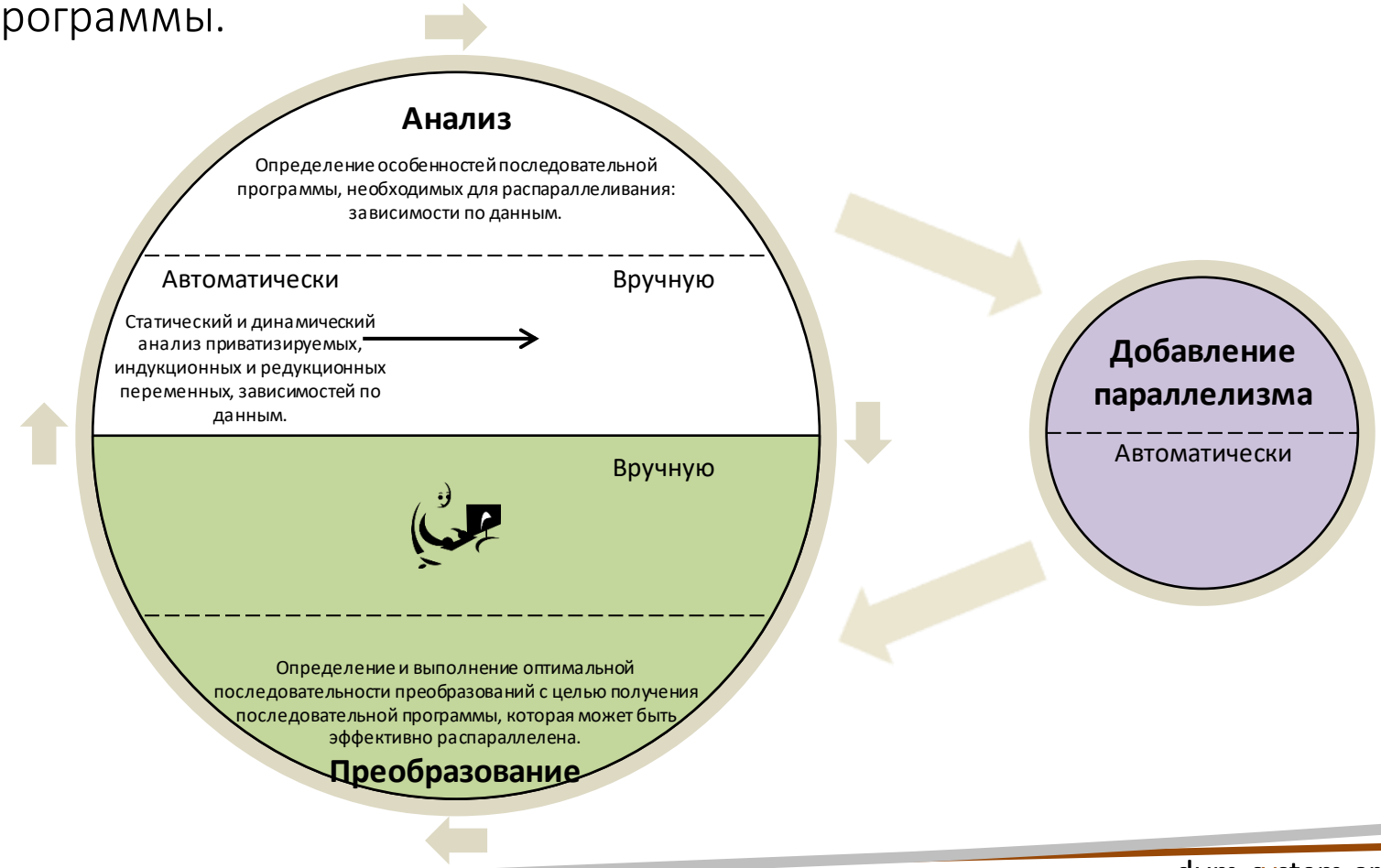
Средства программирования в DVM-системе

C-DVMH = Язык C 99 + спец. прагмы
Fortran-DVMH = Язык Fortran 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 выполнения.

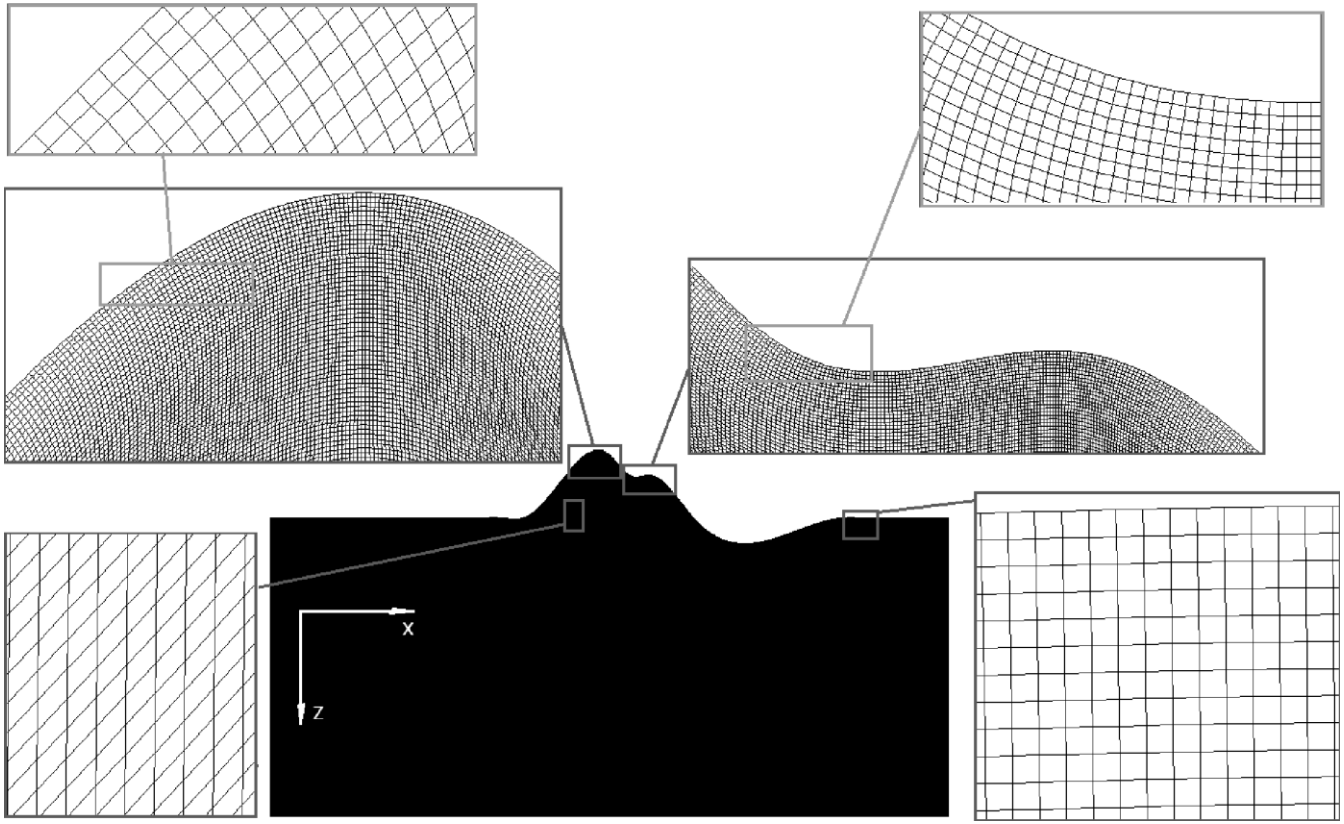
Автоматизация распараллеливания: SAPFOR

- Помогает программисту эффективно отображать его программы на многоядерные кластеры с ускорителями.
- Организует параллельные вычисления полностью автоматически.
- Взаимодействует с программистом в терминах последовательной программы.



Моделирование распространения упругих волн в средах со сложной 3D геометрией

- Решение системы уравнений теории упругости в 3D области.
- Преобразование криволинейной сетки в структурированную: 2D срез



Этапы распараллеливания программы: SAPFOR

- ~ 3000 строк кода на языке Fortran 95, > 100 динамических массивов, > 90 циклов

Visualizer - [m_s_4_double_xy.for]

File Project View Tools Panels Window Help

Project
Source files
m_s_4_double

```
!================================== Inside medium ======================================
!DVM$ REGION

!DVM$ PARALLEL (k,j,i) on amu(i,j,k), SHADOW_RENEW(U(CORNER),
!DVM$& V(CORNER),W(CORNER)), PRIVATE(
!DVM$& dc1Ux_dq1,dc1Ux_dq2,dc1Ux_dq3,dc1Vy_dq1,dc1Vy_dq2,dc1Vy_dq3,
!DVM$& dc1Wz_dq1,dc1Wz_dq2,dc1Wz_dq3,dSxx_dx,dcUy_dq1,dcUy_dq2,
!DVM$& dcUy_dq3,dcVx_dq1,dcVx_dq2,dcVx_dq3,dSxy_dy,dcUz_dq1,dcUz_
!DVM$& dcUz_dq3,dcWx_dq1,dcWx_dq2,dcWx_dq3,dSxz_dz,dSxy_dx,dc
!DVM$& dc2Ux_dq2,dc2Ux_dq3,dc2Vy_dq1,dc2Vy_dq2,dc2Vy_dq3,dc2W
!DVM$& dc2Wz_dq2,dc2Wz_dq3,dSyy_dy,dcVz_dq1,dcVz_dq2,dcVz_d
!DVM$& dcWy_dq1,dcWy_dq2,dcWy_dq3,dSyz_dz,dSxz_dx,dSyz_dy,
!DVM$& dc3Ux_dq2,dc3Ux_dq3,dc3Vy_dq1,dc3Vy_dq3,dc3Wz_
!DVM$& dc3Wz_dq2,dc3Wz_dq3,dSzz_dz)
    do k = k_DOWN, k_UP
    do j = j_DOWN, j_UP
    do i = i_DOWN, i_UP

! dSxx_dx ------------------------------------
    dc1Ux_dq1 =
   & rdd*((0.5d0*alambda(i+1,j,k)+amu(i+1,j,k)+
   &      0.5d0*alambda(i,j,k)+amu(i,j,k))*
   &       dq1_dx(2*i+1,2*j,2*k)*(U(i+1,j,k)-U(i,j,k)) -
   &      (0.5d0*alambda(i,j,k)+amu(i,j,k)+
```

До 53 приватных переменных в одном цикле.

Исследование свойств программы в SAPFOR.

Проверка корректности спецификаций в DVM.

Loop / PU	Progr...	File n...	Level...	Tigtl...	Dependency	... variables	Read variables
k=k_source-1,k_...		m_s_...	1.31.3	0	PRIV(i,j,k);	k	k_source
j=j_source,j_sou...		m_s_...	1.31....	1	PRIV(i,j);	j	j_source
i=i_source,i_sou...		m_s_...	1.31....	1	PRIV(i);	i;u(i,j,k);v(i,j,k);...	i_source;k;k_so...
k=k_source,k_s...		m_s_...	1.31.4	0	PRIV(i,j,k);	k	k_source
j=j_source,j_sou...		m_s_...	1.31....	1	PRIV(i,j);	j	j_source
i=i_source-1,i_s...		m_s_...	1.31....	1	PRIV(i);	i;uu(i,j,k);vv(i,j,...	i_source;i;uu(i,j...
k=k_source,k_s...		m_s_...	1.31.5	0	PRIV(i,j,k);	k	k_source
j=j_source-1,j_s...		m_s_...	1.31....	1	PRIV(i,j);	j	j_source
i=i_source,i_sou...		m_s_...	1.31....	1	PRIV(i);	i;uu(i,j,k);vv(i,j,...	i_source;j;j_sou...
k=k_source-1,k_...		m_s_...	1.31.6	0	PRIV(i,j,k);	k	k_source
j=j_source,j_sou...		m_s_...	1.31....	1	PRIV(i,j);	j	j_source
i=i_source,i_sou...		m_s_...	1.31....	1	PRIV(i);	i;uu(i,j,k);vv(i,j,...	i_source;k;k_so...
i_time=i_time,iti...		m_s_...	1.32	0	PRIV(i_time);OUT+FLOW+ANTI(v,vv,w,ww,u,uu,alambda,amu,ttorhoj,dq1_dx,dq2_dx,dq3_dx,dq1_dy,dq2_dy,dq3_dy,dq1_dz,dq2_dz,dq3_dz,a1,...	i_time	i_time;itime
k=k_down,k_up	com...	m_s_...	2.1	0	PRIV(dcvz_dq2,dcuz_dq3,dc2vy_dq3,dc1vy_dq1,dcvx_dq1,dsyz_dy,dc1wz_dq2,dsxz_dz,dcwy_dq2,dc3wz_dq2,dcwx_dq3,dc2wz_dq3,dc1wz_dq1,d...	k	k_down;k_up
j=j_down,j_up	com...	m_s_...	2.1.1	1	PRIV(dcvz_dq2,dcuz_dq3,dc2vy_dq3,dc1vy_dq1,dcvx_dq1,dsyz_dy,dc1wz_dq2,dsxz_dz,dcwy_dq2,dc3wz_dq2,dcwx_dq3,dc2wz_dq3,dc1wz_dq1,d...	j	j_down;j_up
i=i_down,i_up	com...	m_s_...	2.1.1.1	1	PRIV(dcvz_dq2,dcuz_dq3,dc2vy_dq3,dc1vy_dq1,dcvx_dq1,dsyz_dy,dc1wz_dq2,dsxz_dz,dcwy_dq2,dc3wz_dq2,dcwx_dq3,dc2wz_dq3,dc1wz_dq1,d...	i;dc1ux_dq1;dc...	i_down;i_up;rd...
j=n_down_y,n_...	edges	m_s_...	4.1	0	PRIV(k,r2,dv_dq1,r3,dv_dq2,znam1,dw_dq1,znam2,dw_dq2,znam3,znam4,i,du_dq1,j,r1,du_dq2);OUT+FLOW+ANTI(dv_dq3,dw_dq3,du_dq3);RE...	j	n_down_y;n_u...
k=0,0	edges	m_s_...	4.1.1	1	PRIV(k,r2,dv_dq1,r3,dv_dq2,znam1,dw_dq1,znam2,dw_dq2,znam3,znam4,i,du_dq1,r1,du_dq2);	k	
i=n_down_x,n_u...	edges	m_s_...	4.1.1.1	1	PRIV(r2,dv_dq1,r3,dv_dq2,znam1,dw_dq1,znam2,dw_dq2,znam3,znam4,i,du_dq1,r1,du_dq2);OUT+FLOW+ANTI(dv_dq3,dw_dq3,du_dq3);	i;znam1;znam2...	n_down_x;n_u...

Output | Characteristics

Ready | Ln 1618

Этапы распараллеливания программы: DVM

- Распределение элементов массива между процессорами:

  директивы ***distribute / align***

- Распределение витков цикла между вычислительными устройствами:

  директива ***parallel***

- Организация эффективного доступа к удаленным данным, расположенным на других вычислительных устройствах:

  спецификации ***shadow / across / remote***

- 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***

Распределение данных

Поддержка динамических массивов

```
double precision, allocatable::
      & dq1_dx(:,:,:),dq2_dx(:,:,:),dq3_dx(:,:,:),
      & dq1_dy(:,:,:),dq2_dy(:,:,:),dq3_dy(:,:,:),
      & dq1_dz(:,:,:),dq2_dz(:,:,:),dq3_dz(:,:,:),
        ...
```

Директивы не зависят от количества используемых процессоров

```
!DVM$ DISTRIBUTE dq1_dx(BLOCK, BLOCK, *)
!DVM$ ALIGN (i,j,k) WITH dq1_dx(i,j,k)::dq2_dx, dq3_dx, dq1_dy, dq2_dy
!DVM$ ALIGN (i,j,k) WITH dq1_dx(i,j,k)::dq3_dy, dq1_dz, dq2_dz, dq3_dz
        ...
```

Выделение памяти стандартными операторами языка Fortran

```
        allocate(
      & dq1_dx(2*N_down_x-3:2*N_up_x+3,2*N_down_y-3:2*N_up_y+3,
      & 2*N_down_z-3:2*N_up_z+3),
      & dq2_dx(2*N_down_x-3:2*N_up_x+3,2*N_down_y-3:2*N_up_y+3,
      & 2*N_down_z-3:2*N_up_z+3),
      & dq3_dx(2*N_down_x-3:2*N_up_x+3,2*N_down_y-3:2*N_up_y+3,
      & 2*N_down_z-3:2*N_up_z+3),
        ...)
```

Распределение вычислений

Информация о редукционных и приватных переменных получена от SAPFOR

```
!DVM$ PARALLEL (k,j,i) on amu(i,j,k), PRIVATE(
!DVM$& dc1Ux_dq1,dc1Ux_dq2,dc1Ux_dq3,dc1Vy_dq1,dc1Vy_dq2,dc1Vy_dq3,
!DVM$& dc1Wz_dq1,dc1Wz_dq2,dc1Wz_dq3,dSxx_dx,dcUy_dq1,dcUy_dq2,
        ...)
      do k = k_DOWN, k_UP
      do j = j_DOWN, j_UP
      do i = i_DOWN, i_UP
        ...
```

```
!DVM$ PARALLEL (k,j,i) ON X(2*i,2*j,2*k),
!DVM$& REDUCTION(MINLOC(amax,newSources,3))
      do k=N_down_z,N_up_z
      do j=N_down_y,N_up_y
      do i=N_down_x,N_up_x

      if(((X(2*i,2*j,2*k)-x0)**2 +
     &    (Y(2*i,2*j,2*k)-y0)**2 +
     &    (Z(2*i,2*j,2*k)-z0)**2 < amax)) then
       ...
```

Организация доступа к удаленным данным

Динамический контроль корректности директив с помощью отладчика DVM-системы

./dvm fpdeb M_S_4_double_xy.for

./dvm err M_S_4_double_xy

```
!DVM$ PARALLEL (k,j,i) on amu(i,j,k), PRIVATE(
!DVM$& dc1Ux_dq1,dc1Ux_dq2,dc1Ux_dq3,dc1Vy_dq1,dc1Vy_dq2,dc1Vy_dq3,
!DVM$& dc1Wz_dq1,dc1Wz_dq2,dc1Wz_dq3,dSxx_dx,dcUy_dq1,dcUy_dq2,
       ...)
      do k = k_DOWN, k_UP
      do j = j_DOWN, j_UP
      do i = i_DOWN, i_UP
       dc1Ux_dq1 =
       ...
     & 0.25d0*rdd*( (alambda(i+1,j,k)+2.0d0*amu(i+1,j,k))*
     &  dq2_dx(2*i+2,2*j,2*k)*(U(i+1,j+1,k)-U(i+1,j-1,k)) -
     &                (alambda(i-1,j,k)+2.0d0*amu(i-1,j,k))*
     &  dq2_dx(2*i-2,2*j,2*k)*(U(i-1,j+1,k)-U(i-1,j-1,k)) ) +
       ...
```

*** DYNCONTROL *** : Loop(No(46), Iter(0)), Loop(No(54), Iter(1,2,2)).

Access to non-local element dq2 _dx(2 * i + 2,2 * j,2 * k)

File: M_S_4_double_xy.for Line: 1710

Организация доступа к удаленным данным

```
!DVM$ SHADOW(1:1, 1:1, 1:2) :: U,V,W
!DVM$ SHADOW(2:2, 2:2, 2:2) :: dq1_dx,dq2_dx, dq3_dx, dq1_dy, dq2_dy
!DVM$ SHADOW(2:2, 2:2, 2:2) :: dq3_dy, dq1_dz, dq2_dz, dq3_dz,X,Y,Z
      ...
!DVM$ PARALLEL (k,j,i) on amu(i,j,k), PRIVATE(
!DVM$& dc1Ux_dq1,dc1Ux_dq2,dc1Ux_dq3,dc1Vy_dq1,dc1Vy_dq2,dc1Vy_dq3,
!DVM$& dc1Wz_dq1,dc1Wz_dq2,dc1Wz_dq3,dSxx_dx,dcUy_dq1,dcUy_dq2,
      ...)
      do k = k_DOWN, k_UP
      do j = j_DOWN, j_UP
      do i = i_DOWN, i_UP
       dc1Ux_dq1 =
       ...
     & 0.25d0*rdd*( (alambda(i+1,j,k)+2.0d0*amu(i+1,j,k))*
     &  dq2_dx(2*i+2,2*j,2*k)*(U(i+1,j+1,k)-U(i+1,j-1,k)) -
     &              (alambda(i-1,j,k)+2.0d0*amu(i-1,j,k))*
     &  dq2_dx(2*i-2,2*j,2*k)*(U(i-1,j+1,k)-U(i-1,j-1,k)) ) +
       ...
```

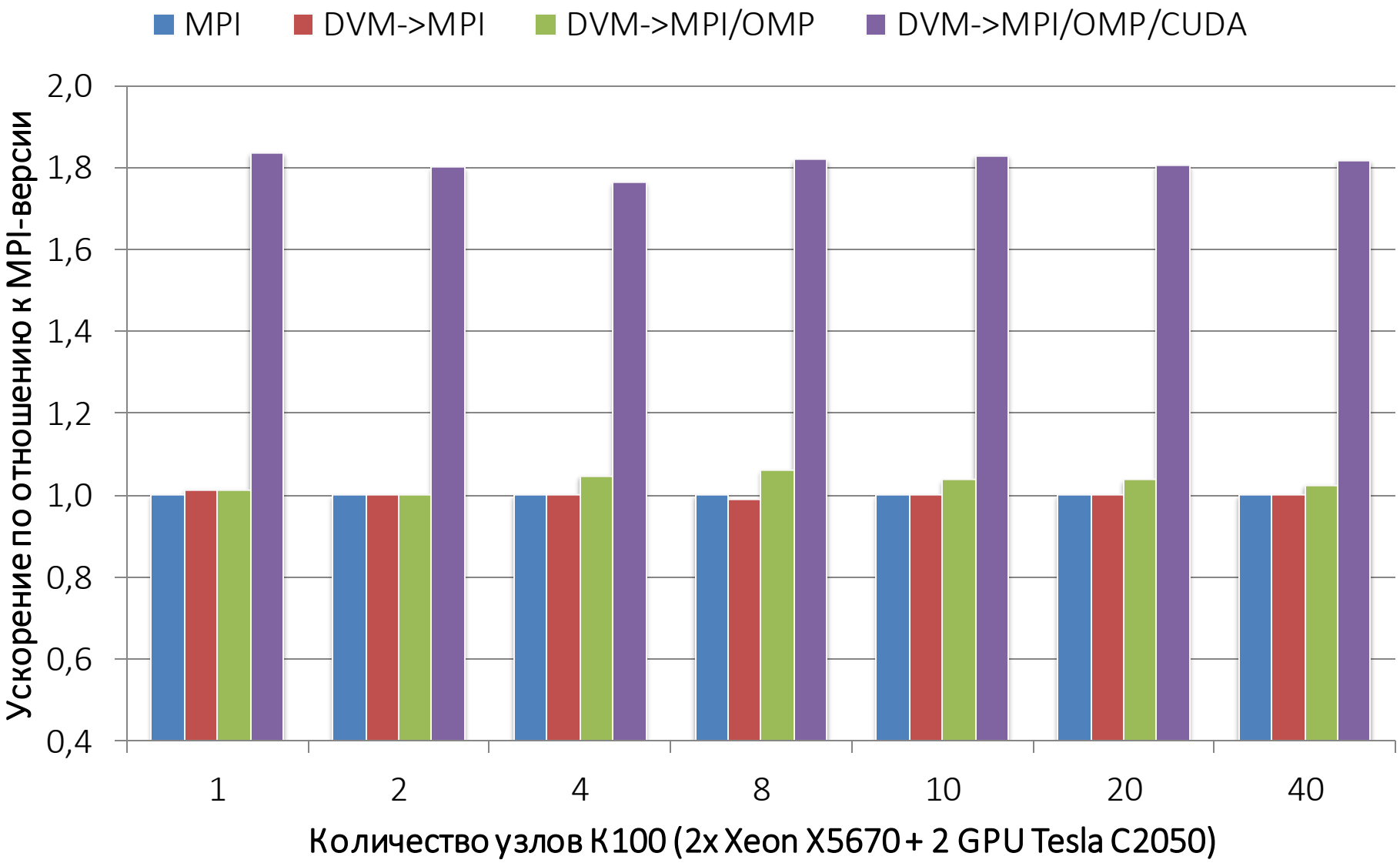
Достаточно увеличить ширину теневой грани до 2 элементов (по умолчанию 1 элемент).

Указание регионов для выполнения на ускорителе

Позволяет выполнять программу на мультипроцессоре, GPU, Xeon Phi.

```
!DVM$ REGION
!DVM$ PARALLEL (k,j,i) on amu(i,j,k), PRIVATE(
!DVM$& dc1Ux_dq1,dc1Ux_dq2,dc1Ux_dq3,dc1Vy_dq1,dc1Vy_dq2,dc1Vy_dq3,
!DVM$& dc1Wz_dq1,dc1Wz_dq2,dc1Wz_dq3,dSxx_dx,dcUy_dq1,dcUy_dq2,
       ...)
      do k = k_DOWN, k_UP
      do j = j_DOWN, j_UP
      do i = i_DOWN, i_UP
       dc1Ux_dq1 =
       ...
     & 0.25d0*rdd*( (alambda(i+1,j,k)+2.0d0*amu(i+1,j,k))*
     &  dq2_dx(2*i+2,2*j,2*k)*(U(i+1,j+1,k)-U(i+1,j-1,k)) -
     &              (alambda(i-1,j,k)+2.0d0*amu(i-1,j,k))*
     &  dq2_dx(2*i-2,2*j,2*k)*(U(i-1,j+1,k)-U(i-1,j-1,k)) ) +
       ...
      end do
      end do
      end do
!DVM$ END REGION
```

Сравнение с ручным распараллеливанием



Выводы

- ~ 3000 строк кода в фиксированном формате на Fortran 95
- > 100 динамических массивов, > 90 циклов

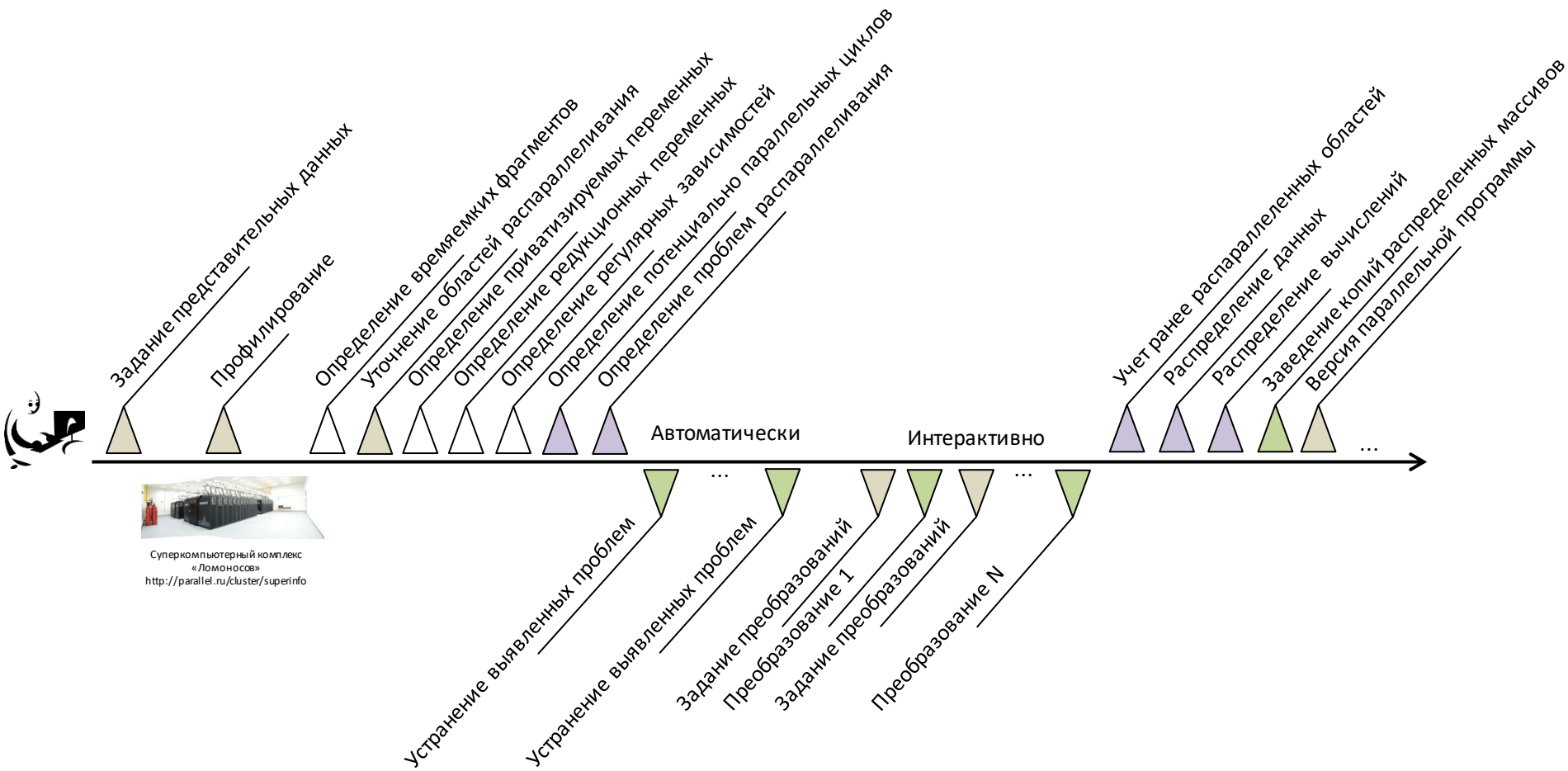
- Было добавлено 79 DVM-директив
- 1 день для распараллеливания и 2 дня на получение результатов
- Параллельное выполнение на гибридном вычислительном кластере с GPU и Xeon Phi
- Всего было задействовано 480 ядер CPU и 80 GPU (40 узлов кластера K100), объем потребляемой памяти при этом составил примерно 1000GB
- Получено практически линейное ускорение задачи:

  слабая и сильная масштабируемости

- Исходный код https://bitbucket.org/dvm-system/elastic-wave-3d
  - SEQ_VER – последовательная версия
  - MPI_VER – параллельная версия с использованием MPI
  - DVMH_VER – параллельная версия в модели DVMH

Архитектура системы САПФОР: будущее

– *итерационное распараллеливание*



Процесс распараллеливания программы – <u>поиск оптимизационной последовательности проходов</u>, обеспечивающей переход от последовательной версии программы к эффективной параллельной версии.

Архитектура системы САПФОР: будущее

Средства профилирования и выделения областей распараллеливания.

Средства поиска последовательностей преобразований для автоматического устранения проблем распараллеливания.

Средства автоматического выполнения наиболее часто встречающихся преобразований.

Средства интерактивного взаимодействия с пользователем в процессе распараллеливания.

Спасибо за внимание

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