Автоматизация преобразований последовательных Фортран-программ, необходимых для их эффективного распараллеливания с помощью системы САПФОР

Катаев Никита

ИПМ им. М.В. Келдыша РАН

Буланов Артем

МГУ им. М.В. Ломоносова

31 марта 2015 г.

Система САПФОР

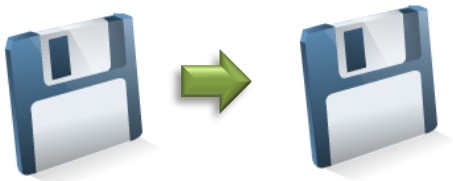
- Помогает программисту эффективно отображать его программы на многоядерные кластеры с ускорителями.

- Организует параллельные вычисления полностью автоматически.
- Взаимодействует с программистом в терминах исходной последовательной программы.
- Помогает программисту преобразовывать последовательные программы для достижения наиболее эффективного распараллеливания.

Трансформация

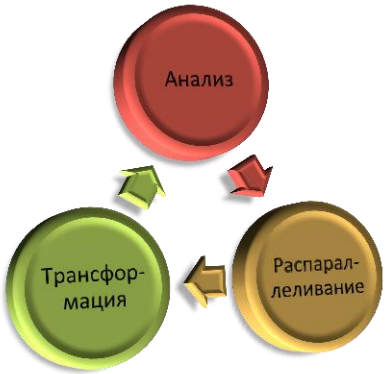
Из последовательной программы в последовательную

- понятно для пользователя – он сможет помочь



Итерационный процесс

- выбор только нужных трансформаций



Итерационный процесс распараллеливания

Visualizer - [example]

File Project View Tools Panels Window Help

Project example
Source files
example.for

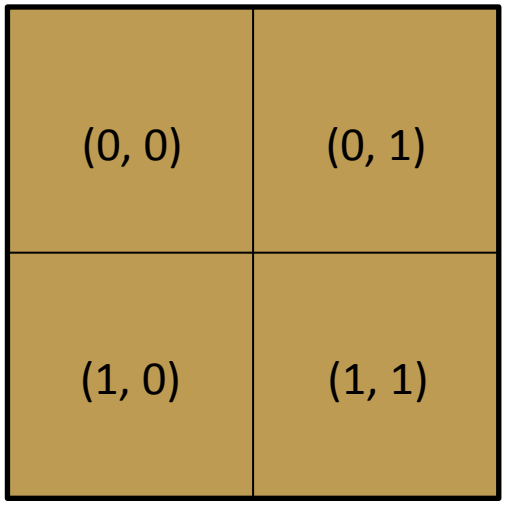
```
PROGRAM EXAMPLE
PARAMETER (N=4)
REAL X(N, 0:N+1)

DO J = 1, N
DO I = 1, N
  X(I,J) = 1.0
ENDDO
ENDDO

DO I = 1, N
  X(I, 0) = -X(L, 1)
  X(I, N+1) = -X(I, N)
ENDDO

PRINT *, X
END
```

Project

```
================Complete==========================
Expert...
D:\YandexDisk\kiam\other\Sys.Sapfor.4.7\Sapfor.DVM\bin\DVM_xp.exe example
Created 4 parallel versions
==================Complete=======================
```

Output

Ready Ln



Система с распределенной памятью

Итерационный процесс распараллеливания

Visualizer - [example]

File Project View Tools Panels Window Help

Project example
Source files
example.for

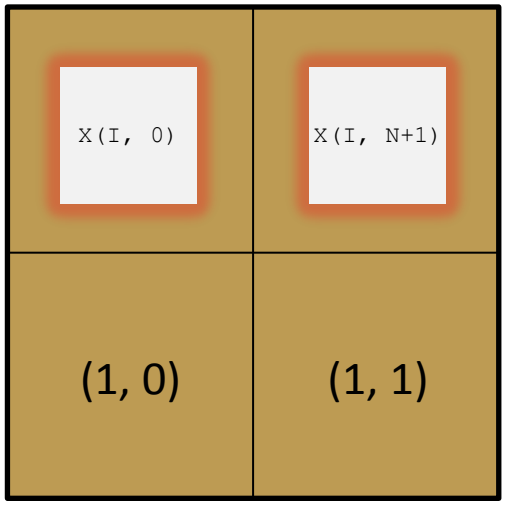
```
PROGRAM EXAMPLE
PARAMETER (N=4)
REAL X(N, 0:N+1)

DO J = 1, N
DO I = 1, N
  X(I, J) = 1.0
ENDDO
ENDDO

DO I = 1, N
  X(I, 0) = -X(I, 1)
  X(I, N+1) = -X(I, N)
ENDDO

PRINT *, X
END
```

Project

Loop / PU	Progr...	File n...	dvm_xp_0 Decisions
i=1,n	exam...	exam...	данные для записи витка x(i,0) и x(i,n+1) находятся на разных процессорах;

Output Decisions

Ready Ln 11



Итерационный процесс распараллеливания

Visualizer - [example_0]

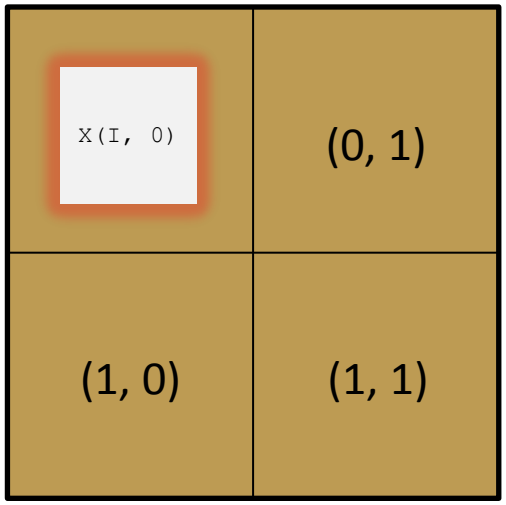
File Project View Tools Panels Window Help

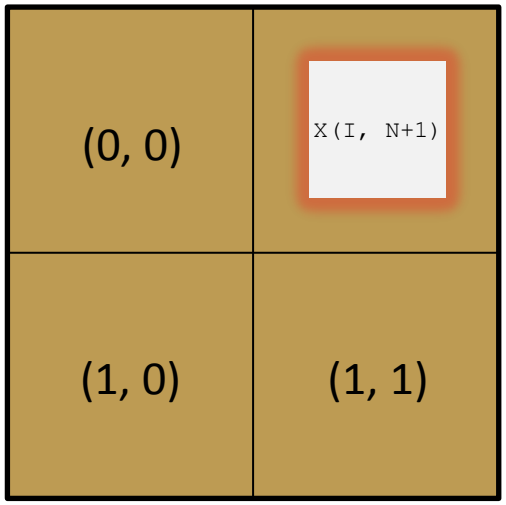
Project example
Source files
example.for

```
      PROGRAM EXAMPLE
      PARAMETER (N=4)
      REAL X(N, 0:N+1)

!DVM$ DISTRIBUTE (BLOCK,BLOCK) :: x
!DVM$ PARALLEL (j,i) ON x(i,j)
      DO J = 1, N
      DO I = 1, N
        X(I, J) = 1.0
      ENDDO
      ENDDO

!DVM$ PARALLEL (i) ON x(i,0),
*DVM$* SHADOW_RENEW (x(0:0,1:1))
      DO I = 1, N
        X(I, 0) = -X(I, 1)
      ENDDO

!DVM$ PARALLEL (i) ON x(i,n+1)
      DO I = 1, N
        X(I, N+1) = -X(I, N)
      ENDDO

      PRINT *, X
      END
```

Project

Ready Ln 1





Три подхода

Автоматический выбор способа устранения проблем

- А если нельзя устранить проблему за одно преобразование?
- А если устранение разных проблем конфликтует?

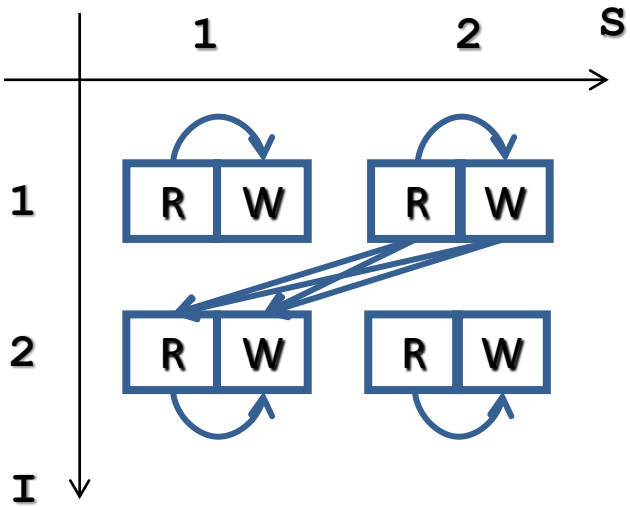
Активное вовлечение пользователя в процесс выбора

- Ручной выбор, автоматическое выполнение.
- Проверка корректности выполняемых преобразований.

Ручное преобразование, если автомат бессилен.

Пример шаблона

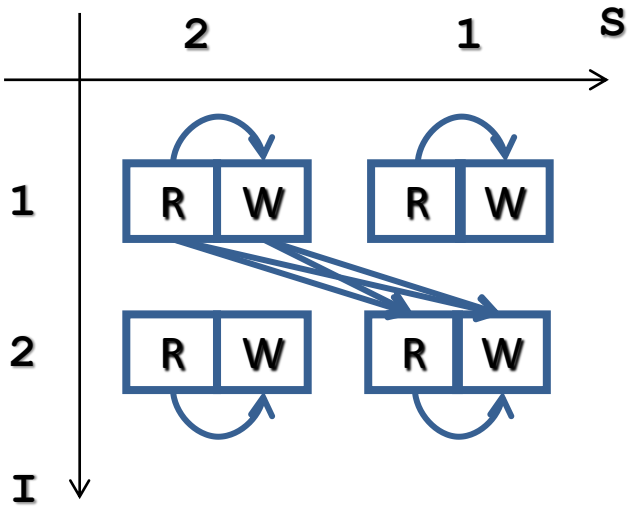
```
do i = 1, n-1
   y = ...
   x(i) = x(i)+y          ! 1
   x(i+1) = x(i+1)-y      ! 2
enddo
```



Разбиение цикла с зависимостями по данным с предварительной перестановкой обращений к массивам.

Пример шаблона

```
do i = 1, n-1
   y = ...
   x(i) = x(i)+y          ! 1
   x(i+1) = x(i+1)-y      ! 2
enddo
```

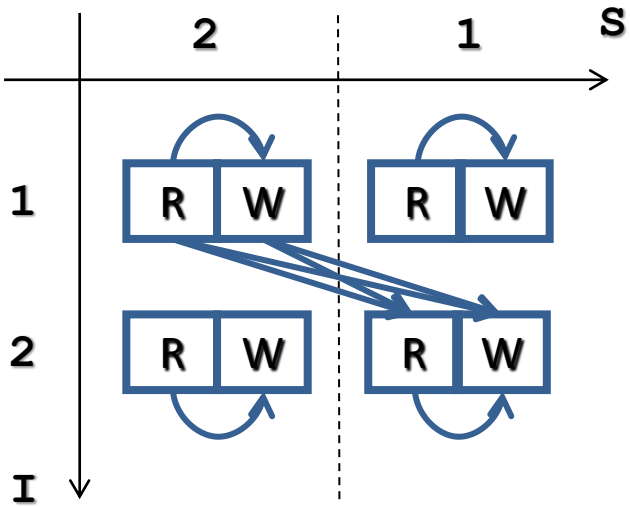


Разбиение цикла с зависимостями по данным с предварительной перестановкой обращений к массивам.

```
do i = 1, n-1
   y = ...
   x(i+1) = x(i+1)-y      ! 2
   x(i) = x(i)+y          ! 1
enddo
```

Пример шаблона

```
do i = 1, n-1
   y = ...
   x(i) = x(i)+y          ! 1
   x(i+1) = x(i+1)-y      ! 2
enddo
```



Разбиение цикла с зависимостями по данным с предварительной перестановкой обращений к массивам.

```
do i = 1, n-1
   y = ...
   y_tmp(i) = y
   x(i+1) = x(i+1)-y
enddo

do i = 1, n-1
   x(i) = x(i)+y_tmp(i)
enddo
```

Помощь пользователя

Диалоговый режим

Специальные комментарии

- Исследования результатов анализа и задания характеристик программы, недоступных для анализа, но необходимых для распараллеливания.
- Исследования предлагаемых вариантов распараллеливания и выбора из них наиболее предпочтительных

Специальный комментарий	Использование
`!PRG INDEPENDENT (<variable-name>)`	Позволяет указать на отсутствие межитерационной зависимостей по данным в цикле.

`!PRG PRIVATE (<variable-name>)`
`!PRG FIRST_PRIVATE (<variable-name>)`
`!PRG LAST_PRIVATE (<variable-name>)`
	Позволяют указать приватные (приватные по входу или по выходу) переменные в цикле.

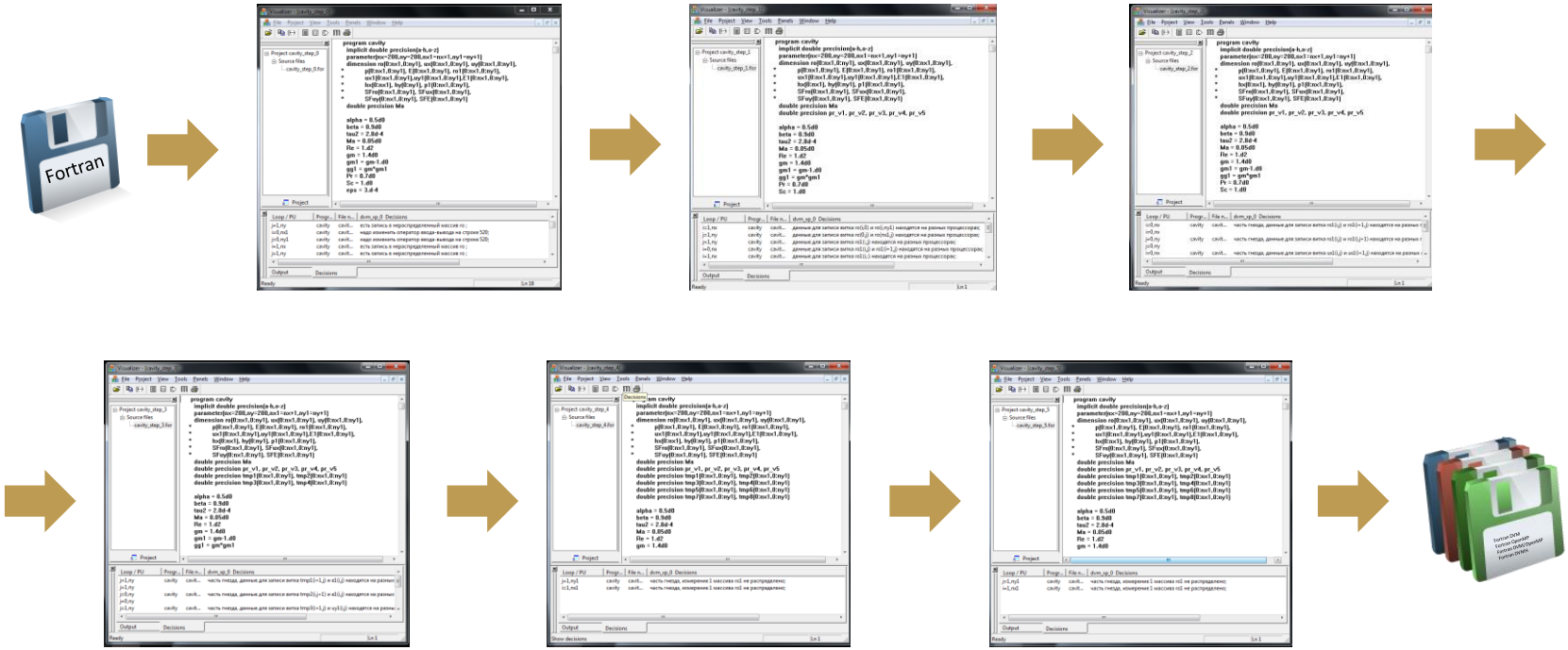
`!PRG FLOW|ANTI|OUTPUT <from> <to> [<len>]`	Позволяет указать на наличие зависимости по данным между двумя операторами и расстояние заданной зависимости.

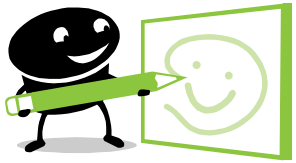
`!PRG REDUCTION (<variable-name> (<op>)`
`[, <array-name>, <index-number>]`
`<op> ::= SUM|PRODUCT|MAX|MIN|AND|OR|EQV|`
`NEQV|MAXLOC|MINLOC`
	Позволяет указать редукционные переменные для цикла. Опциональный параметр <array-name> используется в случае редукции MAXLOC или MINLOC.

Программа «Каверна»

- Моделирование течения несжимаемой жидкости или слабосжимаемого газа около прямоугольной выемки.

- Преобразование программы потребовало выполнения 6 итераций.



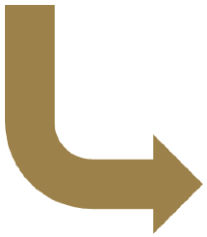


«Каверна»: шаг 1

«надо изменить оператор ввода-вывода на строке 520»

(1 проблема)

```
      do 31 j=1,ny1
        xx = 0.d0
        do 32 i=1,nx1
          write(20,703) xx,yy,RO1(i,j),Ux1(i,j),Uy1(i,j),
     *                  p1(i,j),E1(i,j)
703       format(2f10.6,5e12.4)
32        xx = xx+hx(i)
31      yy = yy+hy(j)
```



```
      do 31 j=1,ny1
        xx = 0.d0
        do 32 i=1,nx1
          pr_v1=RO1(i,j)
          pr_v2=Ux1(i,j)
          pr_v3=Uy1(i,j)
          pr_v4=p1(i,j)
          pr_v5=E1(i,j)
          write(20,703) xx,yy,pr_v1,pr_v2,pr_v3,pr_v4,pr_v5
703       format(2f10.6,5e12.4)
32        xx = xx+hx(i)
31      yy = yy+hy(j)
```

«Каверна»: шаг 2

«данные для записи витка ro(i,0) и ro(i, ny1) находятся на разных процессорах»

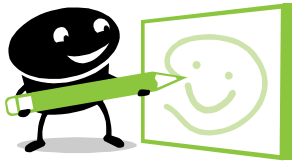
(4 проблемы)

```
 do i = 1,nx
   ro(i,0) = ro(i,1)
   ux(i,0) = -ux(i,1)
   uy(i,0) = -uy(i,1)
   p(i,0) = p(i,1)
   E(i,0) = ro(i,0)*(ux(i,0)*ux(i,0)+uy(i,0)
*          * uy(i,0))*0.5d0 + p(i,0)/gm1
   ro(i,ny1) = ro(i,ny)
   ux(i,ny1) = 2.d0-ux(i,ny)
   uy(i,ny1) = -uy(i,ny)
   p(i,ny1) = p(i,ny)
   E(i,ny1) = ro(i,ny1)*(ux(i,ny1)*ux(i,ny1)
*             + uy(i,ny1)*uy(i,ny1)) * 0.5d0 + p(i,ny)/gm1
  enddo
```

```
do i = 1,nx
   ro(i,0) = ro(i,1)
enddo

 …

do i = 1,nx
    ro(i,ny1) = ro(i,ny)
enddo
```

Для увеличения количества вариантов выравнивания массивов цикл разбивается максимальным образом (на 10 циклов).

4 цикла разбиты на 10 циклов.



«Каверна»: шаг 2

«данные для записи витка ro1(:, j) находятся на разных процессорах»

(5 проблем)

```
do j = 1,ny
     hy4 = hy(j+1)+2.d0*hy(j)+hy(j-1)
     do i = 0,nx
       t = gm*p(i,j)/ro(i,j)
       tp = gm*p(i+1,j)/ro(i+1,j)
       tm = (t+tp)*0.5d0
       …
       ro1(i,j) = ro1(i,j) + Frox
```

```
do j = 1,ny
  do i = 0,nx
    hy4 = hy(j+1)+2.d0*hy(j)+hy(j-1)
      t = gm*p(i,j)/ro(i,j)
      tp = gm*p(i+1,j)/ro(i+1,j)
      tm = (t+tp)*0.5d0
      …
      ro1(i,j) = ro1(i,j) + Frox
```

На итерации внешнего цикла записывается целиком столбец массива, а внутренний цикл гнезда не был распараллелен.

«Каверна»: шаг 3

«данные для записи витка ro1(i, j) и ro(i,j+1) находятся на разных процессорах».

(4 проблемы)

```
do j = 1,ny
  do i = 0,nx
    hy4 = hy(j+1)+2.d0*hy(j)+hy(j-1)
    ...
    Frox = ...
    ro1(i,j) = ro1(i,j) + Frox
    ro1(i+1,j) = ro1(i+1,j) - Frox
    E1(i,j) = E1(i,j) + FEx
    E1(i+1,j) = E1(i+1,j) - FEx
  enddo
enddo
```

```
do j = 1,ny
  do i = 0,nx
    hy4 = hy(j+1)+2.d0*hy(j)+hy(j-1)
    ...
    Frox = ...
    tmp1(i+1,j) = Frox
    ro1(i+1,j) = ro1(i+1,j) - Frox
    E1(i,j) = E1(i,j) + FEx
    E1(i+1,j) = E1(i+1,j) - FEx
  enddo
enddo

do i = 1,nx
  do j = 0, ny
    ro1(i,j) = ro1(i,j) + tmp1(i+1,j)
  enddo
enddo
```

Введены 4 временных массива, 4 цикла разбиты на два.

«Каверна»: шаг 4

«часть гнезда, данные для записи витка tmp1(i+1,j) и e1(i,j) находятся на разных процессорах»

(4 проблемы)

```
do j = 1,ny
  do i = 0,nx
    hy4 = hy(j+1)+2.d0*hy(j)+hy(j-1)
    ...
    Frox = ...
    tmp1(i+1,j) = Frox
    ro1(i+1,j) = ro1(i+1,j) - Frox
    E1(i,j) = E1(i,j) + FEx
    E1(i+1,j) = E1(i+1,j) - FEx
  enddo
enddo

do i = 1,nx
  do j = 0, ny
    ro1(i,j) = ro1(i,j) + tmp1(i+1,j)
  enddo
enddo
```

```
do j = 1,ny
  do i = 0,nx
    hy4 = hy(j+1)+2.d0*hy(j)+hy(j-1)
    ...
    Frox = ...
    tmp1(i+1,j) = Frox
    ro1(i+1,j) = ro1(i+1,j) - Frox
    tmp5(i+1,j) = FEx
    E1(i+1,j) = E1(i+1,j) - Fex
  enddo
enddo

do i = 1,nx
  do j = 0, ny
    E1(i,j) = E1(i,j) + tmp5(i+1,j)
  enddo
enddo

do i = 1,nx
  do j = 0, ny
    ro1(i,j) = ro1(i,j) + tmp1(i+1,j)
  enddo
enddo
```

Введены 4 временных массива, 4 цикла разбиты на два.

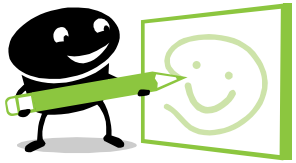
«Каверна»: шаг 5

«часть гнезда, измерение 1 массива ro1 не распределено»

(1 проблема)

```
      do 31 j=1,ny1
        do 32 i=1,nx1
          xx = 0.d0
          pr_v1=RO1(i,j)
          pr_v2=Ux1(i,j)
          pr_v3=Uy1(i,j)
          pr_v4=p1(i,j)
          pr_v5=E1(i,j)
          write(20,703) xx,yy,pr_v1,pr_v2,pr_v3,pr_v4,pr_v5
703       format(2f10.6,5e12.4)
32        xx = xx+hx(i)
31      yy = yy+hy(j)
```

Не влияет на распараллеливание остальной части программы.



«Каверна»: шаг 6

В результате разбиения циклов было добавлено 36 + 4 + 4 = 44 новых цикла.

```
!DVM$ DISTRIBUTE (BLOCK,BLOCK) :: ro
!DVM$ ALIGN ux(i,j) WITH ro(i,j)
!DVM$ ALIGN uy(i,j) WITH ro(i,j)
!DVM$ ALIGN p(i,j) WITH ro(i,j)
!DVM$ ALIGN e(i,j) WITH ro(i,j)
      …
      do i = 1,nx
        ro(i,0) = ro(i,1)
      enddo

      do i = 1,nx
        ux(i,0) = -ux(i,1)
      enddo

      do i = 1,nx
        uy(i,0) = -uy(i,1)
      enddo

      do i = 1,nx
        p(i,0) = p(i,1)
      enddo

      do i = 1,nx
        E(i,0) = ro(i,0)*(ux(i,0)*ux(i,0)+uy(i,0)
     *          * uy(i,0))*0.5d0 + p(i,0)/gm1
      enddo
```

```
do i = 1,nx
  ro(i,0) = ro(i,1)
  ux(i,0) = -ux(i,1)
  uy(i,0) = -uy(i,1)
  p(i,0) = p(i,1)
  E(i,0) = ro(i,0)*(ux(i,0)*ux(i,0)+uy(i,0)
*          * uy(i,0))*0.5d0 + p(i,0)/gm1
enddo
```

Количество добавленных циклов было сокращено до 8.

Заключение

- Итерационный процесс распараллеливания программ позволяет применять только те преобразования, которые необходимы для устранения проблем, препятствующих распараллеливанию.

- Выполняемые перед распараллеливанием преобразования не выводят программу из класса последовательных программ, оставляя ее понятной для пользователя.

Планы

- Увеличение объема поддерживаемых преобразований.

- Совместное применение трех подходов (автомат + полуавтомат + ручной режим) к преобразованию программ.

Спасибо за внимание ☺

?

http://dvm-system.org/