Преобразования последовательных программ при их распараллеливании с помощью системы САПФОР

Н.А. Катаев, М.С. Клинов, Н.В. Поддерюгина

ИПМ им. М.В. Келдыша РАН

САПФОР – система автоматизированного распараллеливания Фортран-программ

- реализация – экспериментальная версия
- промежуточные результаты – эффективное распараллеливание нескольких приложений (тесты NAS BT, LU + 4 приложения ИПМ)

План

- О системе САПФОР
  - состав и схема работы
  - возможности (на фрагментах программ)
  - результаты
- Примеры преобразований
  - типы преобразований
  - объем преобразований на приложениях
- Заключение

Состав системы САПФОР

- Статический анализатор (для языка Fortran)
- Блок распараллеливания
  - кластер (Fortran-DVM)
  - многоядерный кластер (Fortran-DVM/OpenMP)
  - кластер с GPU (Fortran-DVMH)
- Блок создания текста параллельных программ
- Диалоговая оболочка

Ориентация системы на класс задач со статическими регулярными сетками

В данный момент:

- разработка новых алгоритмов анализа
- отладка блоков распараллеливания для кластера и кластера с GPU

Распараллеливание программы

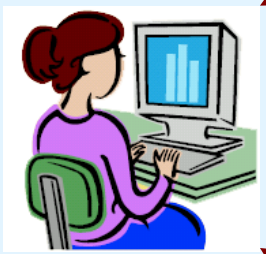
- Преобразование последовательной программы
- Построение разных вариантов распараллеливания
  - вставка директив (DVM, OpenMP) или вызовов (MPI)
  - дополнительные преобразование текста последовательной программы (описание буферов, операции копирования для буферов, условные операторы для MPI-процессов, клонирование процедур)
- Анализ эффективности
  - без запусков параллельной программы (прогноз)
  - с использованием запусков (затраты ресурсов)

Распараллеливание программы в САПФОР

- Преобразование последовательной программы
- Построение разных вариантов распараллеливания
  - вставка директив (DVM, OpenMP) или вызовов (MPI)
  - дополнительных преобразований текста последовательной программы не требуется
- Анализ эффективности
  - без запусков параллельной программы (прогноз)
  - с использованием запусков (затраты ресурсов)

зеленый цвет – делает САПФОР
красный цвет – делает пользователь
черный цвет – не поддерживается или не требуется

Схема работы системы САПФОР



Последовательная программа

Последовательная программа
+ модификации
+ свойства программы

САПФОР

Параллельная программа

Возможности САПФОР (1)

- Фрагмент теста Якоби (решение краевой задачи Дирихле для уравнения Лапласа методом Якоби):

```
do j = 2, n-1
do i = 2, n-1
    b(i,j) =  (a(i-1,j) + a(i,j-1) + a(i+1,j) + a(i,j+1)) / 4
enddo
enddo
```

- Фрагмент теста Cop (решение краевой задачи Дирихле для уравнения Лапласа методом последовательной верхней релаксации):

```
do j = 2, n-1
do i = 2, n-1
    a(i,j) = (w/4) * (a(i-1,j) + a(i+1,j) + a(i,j-1) + a(i,j+1)) + (1-w) * a(i,j)
enddo
enddo
```

Возможности САПФОР (2)

- Фрагмент приложения LU:

```
CPRG private(ue_1jk)
...
do k = 2, nz - 1
do j = 1, ny
do i = 1, nx
        ...
        call exact (1,jglob,k,ue_1jk)
        ...
        do m = 1, 5
              pxi = ( 1.0d+00 - xi ) * ue_1jk(m) + …
              ...
              u(m,i,j,k) = pxi + …
        end do

end do
end do
end do
```

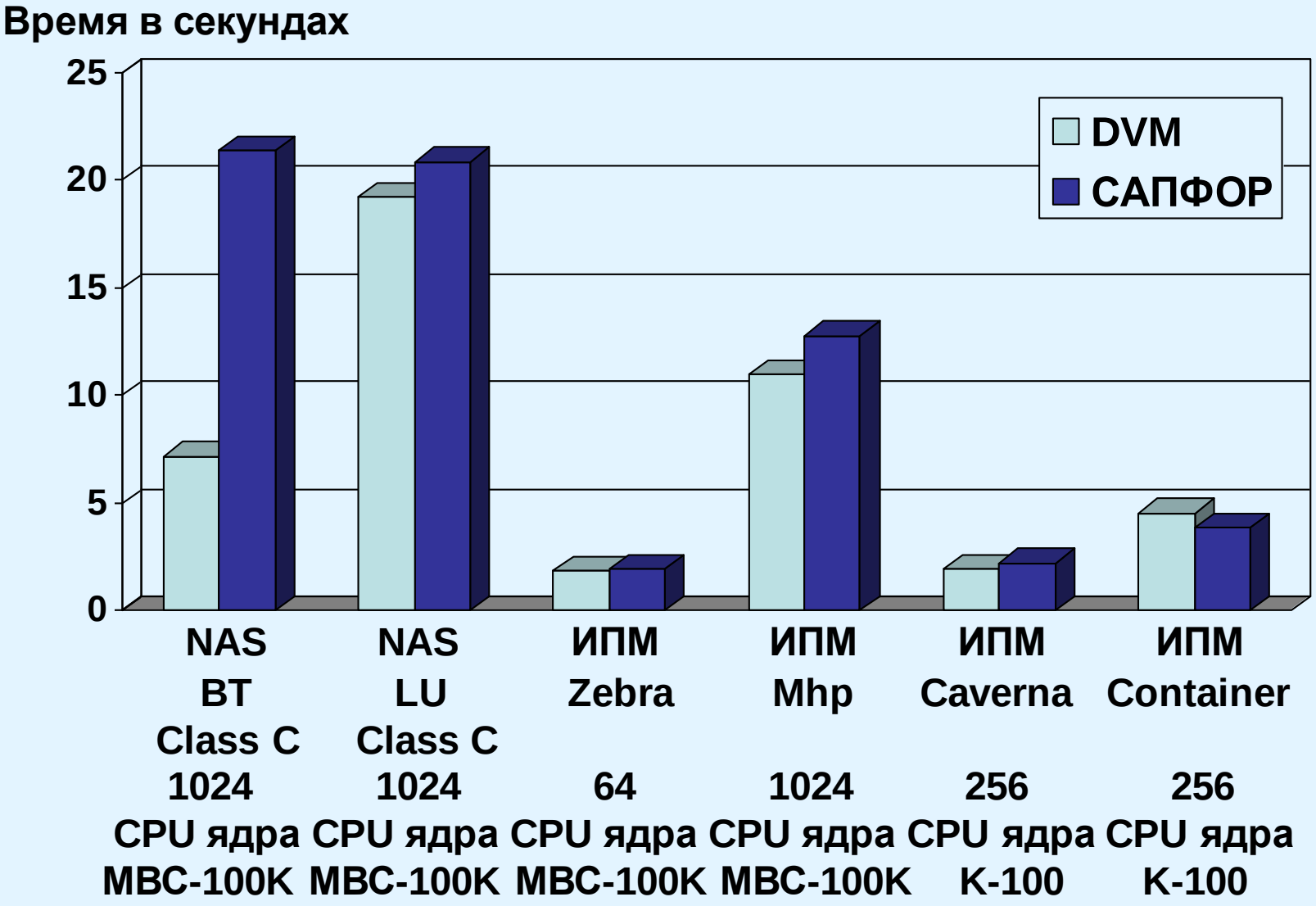
Возможности САПФОР (3)

- Фрагмент приложения Caverna:

```
uu = 0.d0
IDX = -1
CPRG reduction(uu(maxloc),IDX,2)
do j = 1,ny
do i = 1,nx
    …
    c = …
    if(c.gt.uu) then
        uu = c
        IDX(1) = i
        IDX(2) = j
    endif
    …
enddo
enddo

imax = IDX(1)
jmax = IDX(2)
```

Промежуточные результаты САПФОР



План

- О системе САПФОР
  - состав и схема работы
  - возможности (на фрагментах программ)
  - результаты
- Примеры преобразований
  - типы преобразований
  - данные об объеме преобразований на приложениях
- Заключение

Модификации последовательной программы (1)

Разбиение цикла с OUTPUT-зависимостью между витками

```
do j = 1,ny
   do i = 0,nx
      …
      ro1(i,j) = ro1(i,j) + Frox
      ro1(i+1,j) = ro1(i+1,j) - Frox
      …
   enddo
enddo
```

```
dimension tmp1(0:nx1,0:ny1)
dimension tmp2(0:nx1,0:ny1)

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

Модификации последовательной программы (2)

Разбиение цикла с зависимостями OUTPUT и FLOW

```
pbeg = …
do jj = 1,nz
   k = nz+1-jj
   do i = 1,nx
      do j = 1,ny
         …
         p(i,j,k) = pbeg
         …
      enddo
   enddo
   pbeg = pbeg - grav*hz(k)
enddo
```

```
dimension pbeg_arr(nz1)

pbeg = …
pbeg_arr(nz+1)=pbeg
do jj = 1,nz
   k = nz+1-jj
   pbeg = pbeg - grav*hz(k)
   pbeg_arr(k) = pbeg
enddo
do k = 1,nz
   do j = 1,ny
      do i = 1,nx
         …
         p(i,j,k) = pbeg_arr(k+1)
         …
      enddo
   enddo
enddo
```

Модификации последовательной программы (3)

Формирование тесно-вложенных циклов

```
do j = 1,ny
    hy4 = …
    do i = 0,nx
        …
    enddo
enddo
```

```
do j = 1,ny
    do i = 0,nx
        hy4 = …
        …
    enddo
enddo
```

Изменение порядка выполнения витков циклов (полезно и для последовательной программы)

```
do i = 1,nx
    do j = 1,ny
        SFro(i,j) = …
        …
    enddo
enddo
```

```
do j = 1,ny
    do i = 1,nx
        SFro(i,j) = …
        …
    enddo
enddo
```

Модификации последовательной программы (4)

Преобразование циклов с редукцией MAXLOC

```
do j = 1,ny
   do i = 1,nx
      …
      imax = i
      jmax = j
      …
   enddo
enddo
```

```
do j = 1,ny
   do i = 1,nx
      …
      IDX(1) = i
      IDX(2) = j
      …
   enddo
enddo
imax = IDX(1)
jmax = IDX(2)
```

Изменение операторов ввода-вывода

```
write(20,703) xx,yy,RO1(i,j), …
```

```
pr_v1=RO1(i,j)
write(20,703) xx,yy,pr_v1, …
```

Модификации последовательной программы (5)

Разбиение цикла из-за присваивания двух элементов одного массива на одном витке

```
do i = 1,nx
    ro(i,0) = …
    …
    ro(i,ny1) = …
    …
enddo
```

```
do i = 1,nx
    ro(i,0) = …
    …
enddo
do i = 1,nx
    ro(i,ny1) = …
    …
enddo
```

Временное решение для САПФОР - оформление операторов в виде цикла

```
ro(0,0) = …
ux(0,0) = …
uy(0,0) = …
…
```

```
do i = 0,0
    ro(i,0) = …
    ux(i,0) = …
    uy(i,0) = …
    …
enddo
```

Преобразования программиста

Программы	Характеристики	Fortran	Fortran для САПФОР
Приложение ИПМ Caverna	общее число строк	496	554 (111.6%)
	добавлено строк	-	58 (11.6%)
	изменено строк	-	87 (17.5%)
Приложение ИПМ Container	общее число строк	828	864 (104.3%)
	добавлено строк	-	36 (4.3%)
	изменено строк	-	46 (5.5%)

На данный момент имеются данные только для двух программ.
Для других
- либо не подсчитаны (NAS BT LU),
- либо отсутствует изначальная версия программы и подсчитать не представляется возможным (другие 2 приложения ИПМ).

План

- О системе САПФОР
  - состав и схема работы
  - возможности (на фрагментах программ)
  - результаты
- Примеры преобразований
  - типы преобразования
  - объем преобразований на приложениях
- Заключение

Заключение

- Система САПФОР дает хорошие результаты для программ со статическими регулярными сетками (несколько приложений)
- Программист преобразует последовательную программу (последовательная ➔последовательная)
- Программист задает свойства программы, которые не удается определить автоматически (за приемлемое время на имеющихся ресурсах (ЦПУ+ОП+ВП))

- Преобразования в рамках последовательной программы (проще чем преобразования+распараллеливание)
- Набор преобразований достаточно определенный, поэтому можно использовать опыт и нарабатывать навыки
- Некоторые преобразования можно автоматизировать (распознавать и выполнять), на данный момент не реализованы в САПФОР, но планируются