Распараллеливание программных комплексов. Проблемы и перспективы. Система автоматизации распараллеливания SAPFOR

авторы:

А. С. Колганов

В. А. Бахтин, О. Ф. Жукова, Н. А. Катаев,

В. А. Крюков, М. Ю. Кузнецов, Н. В. Поддерюгина,

М. Н. Притула, О. А. Савицкая, А. А. Смирнов

18 сентября 2018г. | Абрау-Дюрсо

Автоматизация распараллеливания

- Распараллеливание программ – процесс их адаптации для эффективного выполнения на вычислительной системе с параллельной архитектурой, который заключается либо в переписывании программ на специальный язык (например, MPI, CUDA, OpenCL), либо во вставке специальной разметки (например, OpenMP, OpenACC, DVM);
- Автоматическое распараллеливание – процесс адаптации программы компиляторами (например, Intel, PGI), состоящий в автоматическом её преобразовании без участия пользователя для эффективного выполнения на параллельной вычислительной системе;
- Автоматизация распараллеливания – процесс адаптации программы, состоящий в автоматизированном ее отображении в параллельную программу, в котором пользователь принимает активное участие. Автоматизация распараллеливания может включать в себя процесс автоматического распараллеливания.

Автоматизация распараллеливания

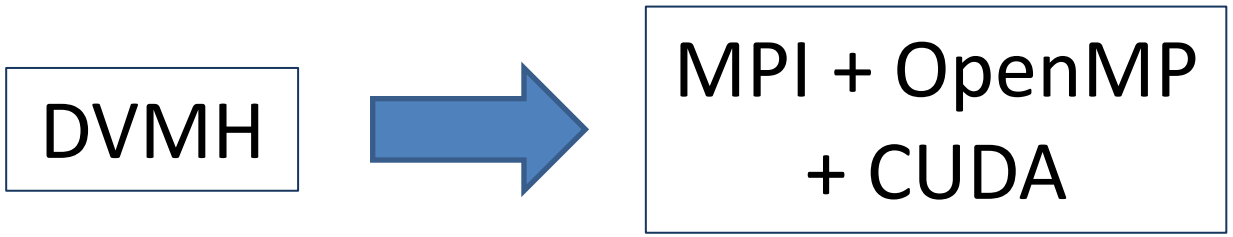
- Распараллеливание программ – процесс их адаптации для эффективного выполнения на вычислительной системе с параллельной архитектурой, который заключается либо в переписывании программ на специальный язык (например, MPI, CUDA, OpenCL), либо во вставке специальной разметки (например, OpenMP, OpenACC, DVM);
- Автоматическое распараллеливание – процесс адаптации программы компиляторами (например, Intel, PGI), состоящий в автоматическом её преобразовании без участия пользователя для эффективного выполнения на параллельной вычислительной системе;
- Автоматизация распараллеливания – процесс адаптации программы, состоящий в автоматизированном ее отображении в параллельную программу, в котором пользователь принимает активное участие. Автоматизация распараллеливания может включать в себя процесс автоматического распараллеливания.

Языки и системы параллельного программирования

- Низкоуровневые – **MPI**, SHMEM, pThread, TBB,**CUDA**, OpenCL;
- Высокоуровневые – **OpenMP**, OpenACC, **DVMH**, HPF, CoArray Fortran, UPC, Titanium,Chapel, X10, Fortress, XcalableACC, XcalableMP;
- Системы автоматизации – CAPTools, Parawise, FORGE Magic/DM, BERT77, ParalWare Trainer, Appolo, **SAPFOR**, Plutto;

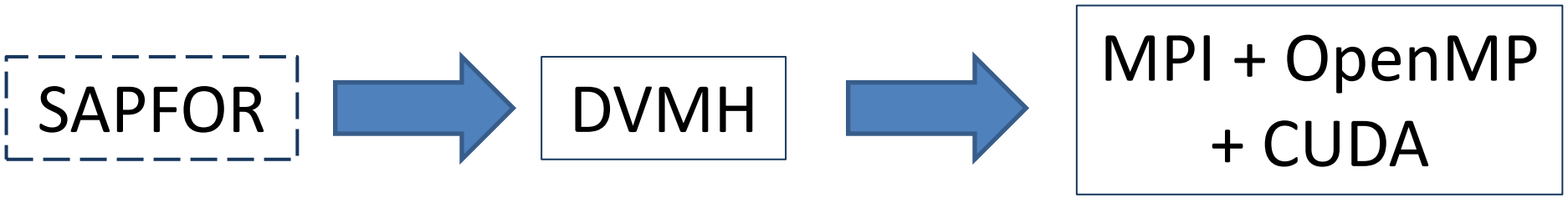
Языки и системы параллельного программирования

- Низкоуровневые – **MPI**, SHMEM, pThread, TBB,**CUDA**, OpenCL;
- Высокоуровневые – **OpenMP**, OpenACC, **DVMH**, HPF, CoArray Fortran, UPC, Titanium,Chapel, X10, Fortress, XcalableACC, XcalableMP;
- Системы автоматизации – CAPTools, Parawise, FORGE Magic/DM, BERT77, ParalWare Trainer, Appolo, **SAPFOR**, Plutto;



Языки и системы параллельного программирования

- Низкоуровневые – **MPI**, SHMEM, pThread, TBB,**CUDA**, OpenCL;
- Высокоуровневые – **OpenMP**, OpenACC, **DVMH**, HPF, CoArray Fortran, UPC, Titanium,Chapel, X10, Fortress, XcalableACC, XcalableMP;
- Системы автоматизации – CAPTools, Parawise, FORGE Magic/DM, BERT77, ParalWare Trainer, Appolo, **SAPFOR**, Plutto;



Средства программирования в DVM-системе

C-DVMH = Язык C 99 + спец. прагмы
Fortran-DVMH = Язык Fortran 95 + спец. комментарии

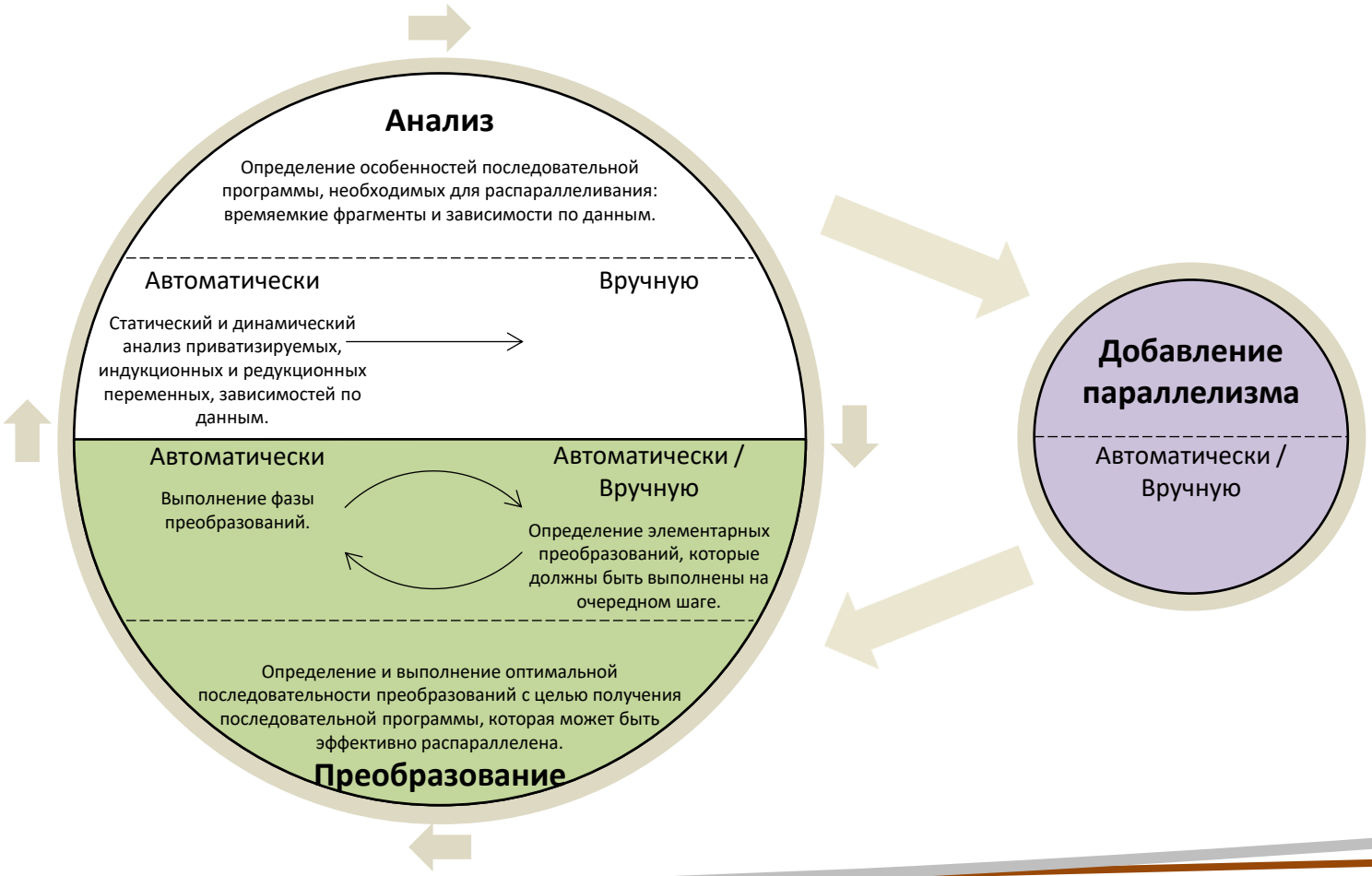
- Специальные комментарии и прагмы являются высокоуровневыми спецификациями параллелизма в терминах последовательной программы.
- Отсутствуют низкоуровневые передачи данных и синхронизации в коде программы.
- Последовательный стиль программирования.
- Спецификации параллелизма «невидимы» для стандартных компиляторов.
- Существует единственный экземпляр программы для последовательного и параллельного выполнения.

```
PROGRAM    JACOBY_DVMH
           PARAMETER    (L=4096,  ITMAX=100)
           REAL     A(L,L), B(L,L)
!DVM$      DISTRIBUTE    ( BLOCK,   BLOCK)   ::   A
!DVM$      ALIGN  B(I,J)  WITH  A(I,J)
           PRINT *,  '**********  TEST_JACOBI   **********'
           DO IT  =  1,  ITMAX
!DVM$         REGION
!DVM$            PARALLEL (J,I) ON  A(I,  J)
                 DO  J  =  2, L-1
                      DO  I  =  2, L-1
                          A(I, J)  =  B(I, J)
                      ENDDO
                 ENDDO
!DVM$            PARALLEL (J,I) ON B(I,  J), SHADOW_RENEW (A)
                 DO  J = 2,  L-1
                       DO  I = 2,  L-1
                            B(I, J) =  (A(I-1, J) + A(I, J-1) +  A(I+1, J) + A(I, J+1)) / 4
                       ENDDO
                  ENDDO
!DVM$      END REGION
           ENDDO
!DVM$      GET_ACTUAL(B)
           PRINT *,B
           END
```

Алгоритм Якоби в модели DVMH

Автоматизация распараллеливания: SAPFOR

- Помогает программисту эффективно отображать его программы на многоядерные кластеры с ускорителями;
- Взаимодействует с программистом в терминах последовательной программы.



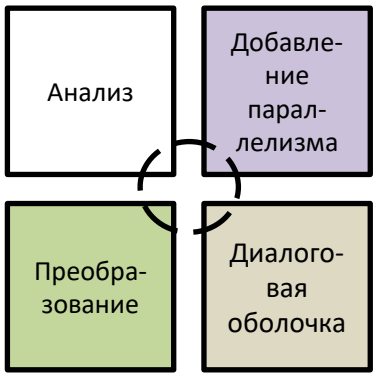
Автоматизация распараллеливания: SAPFOR

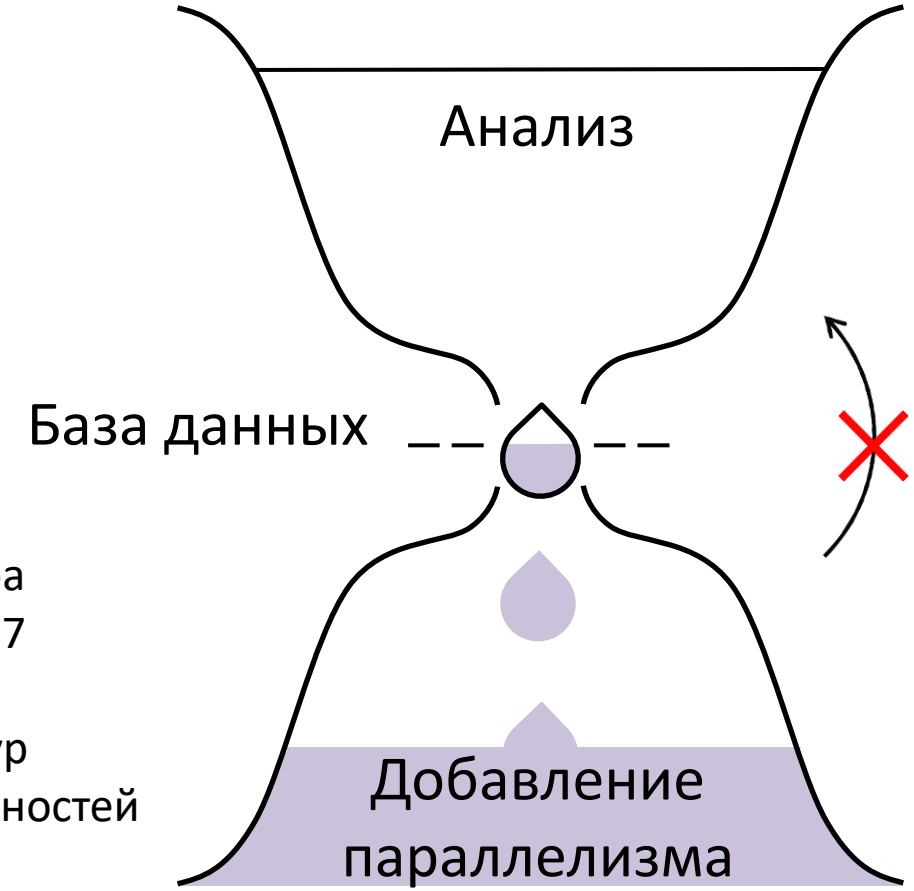
- Создана в Институте прикладной математики им. М.В. Келдыша РАН в 2009 году;
- Аббревиатура САПФОР – система автоматизации распараллеливания последовательных Фортран-программ;
- Состояла из двух основных подсистем: статического анализатора и эксперта (распределение данных и вычислений, доступ к удаленным данным, генерация параллельной FDVM-программы);
- Помогает программисту эффективно отображать его программы на многоядерные кластеры с ускорителями;
- Взаимодействует с программистом в терминах последовательной программы.

Архитектура системы SAPFOR: прошлое

- **Ограниченная** поддержка итерационного распараллеливания;
- **Невозможность** распараллеливания только части программы препятствует распараллеливанию сложных комплексов.

Набор независимых компонент





- Контекстно-независимая структура
- Ориентированность на Фортран 77
- Огрубление информации
- Требование подстановки процедур
- Неполное использование возможностей FDVMH-языка
- Отсутствие каких-либо преобразований

Проблемы распараллеливания последовательной программы на кластер

- Многомодульность, многовариантость, многоязыковость;

- Отображение данных на узлы кластера:
  - размножение данных;
  - распределение данных;

- Выбор наиболее оптимального / эффективного способа распределения данных => перебор всех вариантов?

Проблема многомодульности

- Процедура используется как для работы с данными, которые распределены по узлам кластера, так и с данными, которые находятся на каждом узле кластера;
- Функция выполняется как на центральном процессоре, так и на ускорителе;
- **Подстановка процедур или клонирование?**
  + сокращение накладных расходов на вызов процедур/функций
  + улучшение производительности за счет выполнения дополнительних оптимизаций после подстановки, учитывая контекст вызываемой и вызывающей процедуры/функции
  - невозможность подстановки для программ, написанных на разных языках программирования (например, Фортран и Си)
  - надежда на «успешность» ряда оптимизаций, которые могут быть выполнены после подстановки, может не оправдаться
- **Требуется определить критерии, определяющие необходимость подстановки процедур/функций;**
- **Требуется реализовать проход, выполняющий необходимые подстановки и клонирование процедур/функций.**

Проблема многовариантности

Современные программные комплексы вычислительного характера имеют сложную структуру и могут включать в себя следующие компоненты:

- Средства для постановки задачи, которые используются для построения геометрической модели физической расчетной области, генерации расчетных сеток, определения начальных и граничных условий;
- Вычислительное ядро, с помощью которого осуществляется расчет;
- Средства для сохранения, обработки и визуализации результатов расчетов.

Для решения систем уравнений в вычислительном ядре могут быть реализованы различные солверы.

Ядро может быть универсальным, выполнять как 2D, так и 3D расчеты, поддерживать различные типы дискретизации, состоять из множества частей.

При проведении модельных расчетов многие части комплекса могут не выполняться, но их распараллеливание может требовать принятия решений, которые противоречат требованиям других частей программы.

Задача поиска оптимального распределения данных и вычислений может стать неразрешимой => требуется реализовать **режим инкрементального распараллеливания**

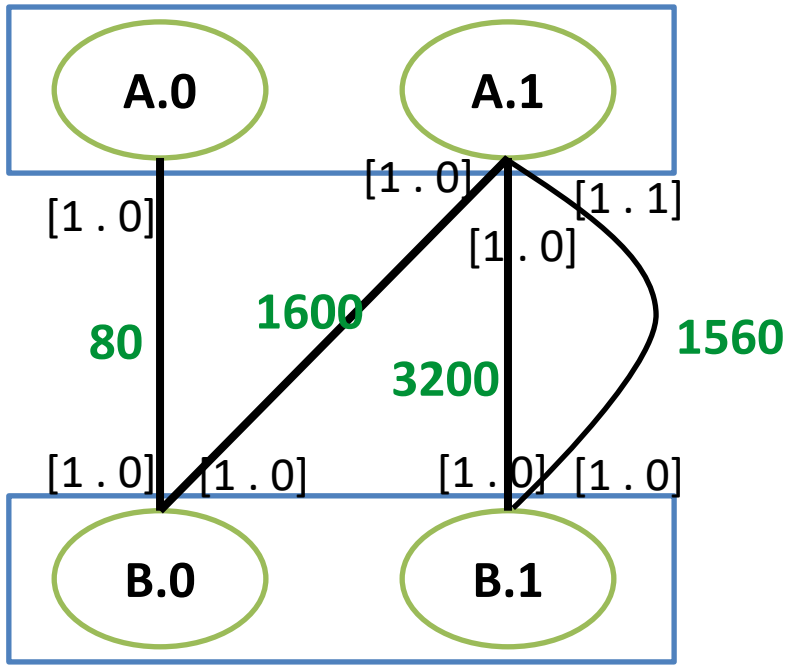
Проблема многоязыковости

- разные модули могут быть написаны на разных языках программирования, например, Fortran, C или C++;
- невозможность выполнить подстановку процедур, написанных на одном языке программирования, в исходный код на другом языке;
- Использование внутренних структур системы SAPFOR для абстрагирования от языка программы.

Детальный дизайн системы SAPFOR: распределение данных

- Отображение обращений к массивам в виде **графа массивов**;
- Рассматриваются не обязательно тесно гнездовые циклы;
- Рассматриваются только линейные индексные выражения **a*K + b**, K – переменная цикла;

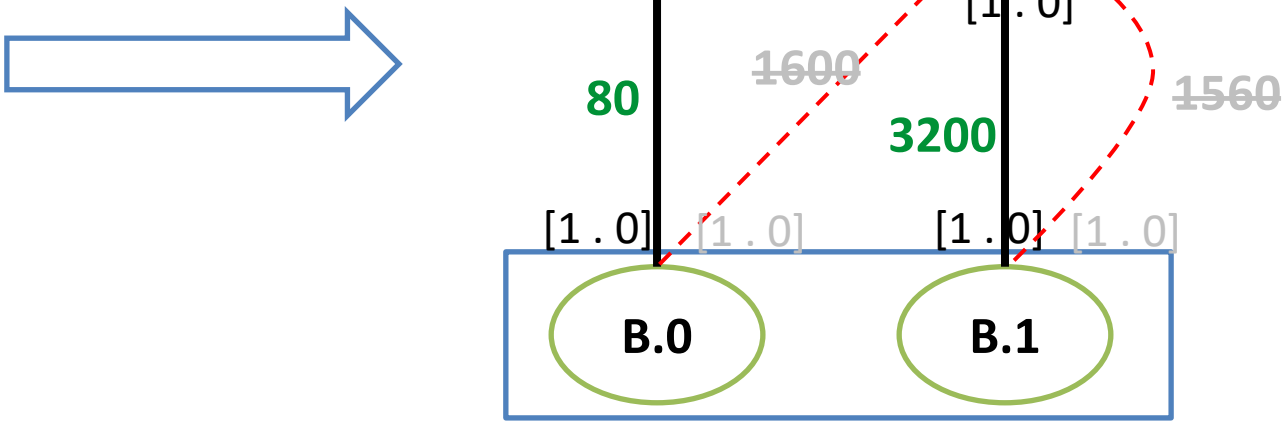
```
DO I = 1, 40
   Z = A(I, 1)
   DO K = 1, 40
     A(I, K) = B(I, K) + B(K,K) / Z
   ENDDO
ENDDO

DO I = 1, 40
   Z = A(I, 2)
   DO K = 1, 39
     A(I, K) = B(I, K + 1) + Z
   ENDDO
ENDDO
```



Детальный дизайн системы SAPFOR: распределение данных

- Разрешение конфликтов **графе массивов**:
  - Поиск простых циклов;
  - Поиск набора дуг с минимальным совокупным весом;
- Построение распределения данных по графу массивов.

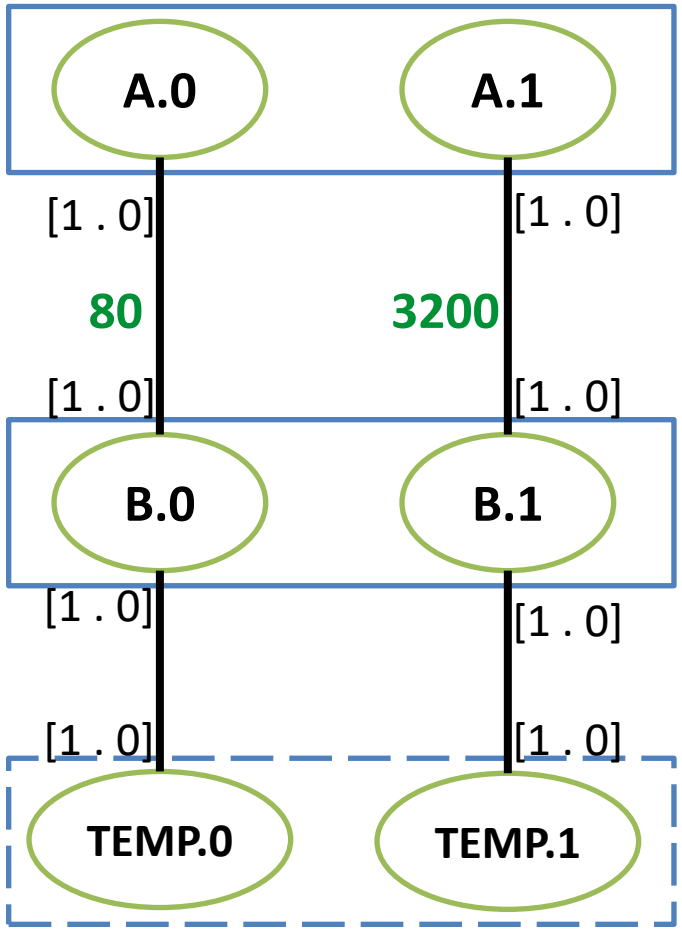
```
DO I = 1, 40
   Z = A(I, 1)
   DO K = 1, 40
     A(I, K) = B(I, K) + B(K,K) / Z
   ENDDO
ENDDO

DO I = 1, 40
   Z = A(I, 2)
   DO K = 1, 39
     A(I, K) = B(I, K + 1) + Z
   ENDDO
ENDDO
```



Детальный дизайн системы SAPFOR: распределение вычислений

- Создание варианта распределения данных;
- Построение параллельной версии программы на основе анализа AST и графа массивов.

```
!DVM$  DISTRIBUTE TEMP(BLOCK, BLOCK)
!DVM$  ALIGN(i,j) with TEMP(i,j) :: A, B
!DVM$  PARALLEL (I,K) on TEMP(I,K), REMOTE_ACCESS(A(I,1),
!DVM$& B(K, K))
DO I = 1, 40
   DO K = 1, 40
     Z = A(I, 1)
     A(I, K) = B(I, K) + B(K,K) / Z
   ENDDO
ENDDO
!DVM$  PARALLEL (I,K) on TEMP(I,K), REMOTE_ACCESS(A(I,2)),
!DVM$&  SHADOW_RENEW(B)
DO I = 1, 40
   DO K = 1, 39
     Z = A(I, 2)
     A(I, K) = B(I, K + 1) + Z
   ENDDO
ENDDO
```



Инкрементальное распараллеливание на кластер

- Область распараллеливания – область последовательной программы с одним входом и одним выходом, которая ограничивает действие SAPFOR во всей программе;

- Области распараллеливания позволяют производить **инкрементальное распараллеливание на кластер** с помощью **DVM-системы**;

- Достоинства данного подхода:
  - Возможность **распараллеливания не всей программы, а ее времяемких фрагментов**. Это упрощает работу системы SAPFOR и программиста, так как существенно сокращается объем кода программы для анализа и распараллеливания;
  - Возможность **использования найденных решений для времяемких фрагментов** в качестве подсказки при исследовании оставшихся частей программы на следующих итерациях распараллеливания в системе;
  - Возможность **ручного распараллеливания некоторых фрагментов программы** и учета принятых программистом решений при распараллеливании других фрагментов системой SAPFOR.

Инкрементальное распараллеливание на кластер

- Область распараллеливания – область последовательной программы с одним входом и одним выходом, которая ограничивает действие SAPFOR во всей программе;

```
          DO IT  =  1,  ITMAX
!$SPF PARALLEL_REG   firstLoop
             DO  J  =  2, L-1
                DO  I  =  2, L-1
                    A(I, J)  =  B(I, J)
                ENDDO
             ENDDO
!$SPF END PARALLEL_REG

!$SPF PARALLEL_REG   secondLoop
             DO  J = 2,  L-1
                DO  I = 2,  L-1
                   B(I, J) =  (A(I-1, J) + A(I, J-1) +  A(I+1, J) + A(I,J+1)) / 4
                ENDDO
             ENDDO
          ENDDO
!$SPF END PARALLEL_REG
```

Опыт применения областей распараллеливания на примере BT исходной версии

Шаг 1. Анализ и профилирование

Опыт применения областей распараллеливания на примере BT исходной версии

Шаг 1. Анализ и профилирование

Название процедуры	Время выполнения, сек	Количество строк кода
Вся программа	696 (100%)	3200 (100%)
X_SOLVE	181.06 (26%)	320 (10%)
Y_SOLVE	212.85 (30.5%)	320 (10%)
Z_SOLVE	227.36 (32.66%)	320 (10%)
COMPUTE_RHS	70.5 (10.14%)	306 (9.5%)
ADD	4.7 (0.68%)	25 (0.78%)

Опыт применения областей распараллеливания на примере BT исходной версии

Шаг 2. Выделение необходимого кода в области распараллеливания

Название процедуры	Время выполнения, сек	Количество строк кода
Вся программа	696 (100%)	3200 (100%)
X_SOLVE	181.06 (26%)	320 (10%)
Y_SOLVE	212.85 (30.5%)	320 (10%)
Z_SOLVE	227.36 (32.66%)	320 (10%)
COMPUTE_RHS	70.5 (10.14%)	306 (9.5%)
ADD	4.7 (0.68%)	25 (0.78%)

40 % кода
>90 % времени

Опыт применения областей распараллеливания на примере BT исходной версии

Шаг 3. Распараллеливание системой SAPFOR, разрешение конфликтов и ошибок

- Необходима подстановка процедур, увеличение объема кода на 75%, время выполнения увеличилось на 7%;
- Системой SAPFOR было вставлено 74 DVMH-директивы;
- Хорошая масштабируемость на общей памяти (80% эффективности).

Опыт применения областей распараллеливания на примере BT исходной версии

Шаг 4. Объединение областей распараллеливания, получение эффективной программы (время в сек)

Процессоров / узлов	**1/1**	**16/1**	**64/4**	**256/16**
SAPFOR исходная	**917**	**169.1 (84,8 + 84,3)**	**120.05 (38,25 + 81,8)**	**RTS ERROR**
SAPFOR потенц. пар.	857	84,6	28,4	12,6
MPI	1094	108,8	24,9	11,5

Опыт применения областей распараллеливания на примере BT исходной версии

Шаг 5. Реализация преобразований программы для устранения конфликтов распределения данных, повышение эффективности программы.

Процессоров / узлов	**1/1**	**16/1**	**64/4**	**256/16**
SAPFOR исходная	**917**	**169.1 (84,8 + 84,3)**	**120.05 (38,25 + 81,8)**	**RTS ERROR**
SAPFOR потенц. пар.	857	84,6	28,4	12,6
MPI	1094	108,8	24,9	11,5

Архитектура системы SAPFOR: настоящее

- Распараллеливание программы представляет собой последовательность проходов анализа и преобразований;
- Введены области распараллеливания для обеспечения инкрементального распараллеливания;
- Существенно расширен класс распараллеливаемых программ за счет пересмотра алгоритмов распределения данных и вычислений;
- Использование AST для анализа кода.