ВСЕРОССИЙСКАЯ НАУЧНАЯ КОНФЕРЕНЦИЯ
НАУЧНЫЙ СЕРВИС В СЕТИ ИНТЕРНЕТ

Развитие метода сравнительной отладки DVMH-программ

А.А. Ермичев,
В.А. Крюков

План доклада

1. Сравнительная отладка
2. DVM-система
3. Текущая реализация отладки в DVM-системе
4. Развитие сравнительной отладки DVMH-программ

Отладка параллельных программ

Проблемы:

1. Параллельные алгоритмы сложнее последовательных
2. Нетипичные для последовательной отладки ошибки
3. Недетерминированное поведение отлаживаемой программы
4. Большой объем вычислений

- Использование ручных методов отладки сильно ограничено

- **Автоматизированные методы отладки**

Сравнительная отладка

Основной принцип: Сравнение процесса выполнения двух вариантов программы, контроль промежуточных результатов в определенных контролируемых точках (заданных вручную или автоматически).

Варианты реализации:

1. Сохранение отладочной информации в файл трассы
    a. Сравнение результатов с трассой во время работы второй программы
    b. Сравнение двух файлов трасс
2. Обмен информацией между работающими программами

- Хорошо подходит для процесса распараллеливания
    - Последовательная программа используется как эталон

DVM-система

- DVMH – модель, объединяющая достоинства параллелизма по данным и параллелизма по управлению
- Высокоуровневые языки программирования **C-DVMH** и **Fortran-DVMH**, предоставляющие директивы и управляющие комментарии для создания параллельных алгоритмов
  - Runtime-библиотека LibDVMH
  - DVMH-отладчик
  - Анализатор производительности
- Универсальный исходный код программы для работы на кластере, графических и иных ускорителях

DVM-система – пример

```
double precision a(n, n), b(n, n)

<...>

do while ((it <= itmax).and.(eps > maxeps))
  eps = 0.D0


  do j = 2, n - 1
    do i = 2, n - 1
      b(i, j) = (a(i-1, j) + a(i+1, j) + a(i, j-1) + a(i, j+1)) / 4.D0
    enddo
  enddo


  do j = 2, n - 1
    do i = 2, n - 1
      eps = max(eps, abs(a(i, j) - b(i, j)))
      a(i, j) = b(i, j)
    enddo
  enddo

  it = it + 1
enddo
```

DVM-система – пример

```
      double precision a(n, n), b(n, n)
!dvm$ distribute(block, block) :: a
!dvm$ align b(i,j) with a(i, j)

      <...>

      do while ((it <= itmax).and.(eps > maxeps))
        eps = 0.D0

!dvm$   parallel(j, i) on a(i, j), shadow_renew(a)
        do j = 2, n - 1
          do i = 2, n - 1
            b(i, j) = (a(i-1, j) + a(i+1, j) + a(i, j-1) + a(i, j+1)) / 4.D0
          enddo
        enddo

!dvm$   parallel(j, i) on a(i, j), reduction(max(eps))
        do j = 2, n - 1
          do i = 2, n - 1
            eps = max(eps, abs(a(i, j) - b(i, j)))
            a(i, j) = b(i, j)
          enddo
        enddo

        it = it + 1
      enddo
```

Сравнительная отладка в DVM-системе

- Результат выполнения эталонной программы записывается в файл трассы, а затем подается на вход отлаживаемому алгоритму.
- Автоматическая расстановка контролируемых точек:
  - чтения и записи переменных
  - начало, конец и итерации циклов
- Опции отладки, позволяющие значительно сократить объем трассы без ущерба покрытию операторов отлаживаемой программы:
  - Метод интегральных характеристик массива
  - Метод граничных итераций

Пример – файл трассы

```
BW: [4] "eps"; {"test.f", 26}
AW: [4] "eps" = 0; {"test.f", 26}
PL: 2() [2]; {"test.f", 29}, 96.B
  IT: 18, (2,2)
  BW: [4] "b(i,j)"; {"test.f", 31}
  RD: [4] "a(i - 1,j)" = 681.72562479972839; {"test.f", 31}
  RD: [4] "a(i + 1,j)" = 551.3431462012436; {"test.f", 31}
  RD: [4] "a(i,j - 1)" = 681.72562479972839; {"test.f", 31}
  RD: [4] "a(i,j + 1)" = 551.3431462012436; {"test.f", 31}
  AW: [4] "b(i,j)" = 616.534385500486; {"test.f", 31}
  <...>
EL: 2; {"test.f", 33}, 96.E
PL: 3() [2]; {"test.f", 36}, 97.B
  IT: 18, (2,2)
  RV_BW: [4] "eps"; {"test.f", 38}
  RD: [4] "a(i,j)" = 616.51675929759654; {"test.f", 38}
  RD: [4] "b(i,j)" = 616.534385500486; {"test.f", 38}
  RV_AW: [4] "eps" = 0.017626202889459819; {"test.f", 38}
  BW: [4] "a(i,j)"; {"test.f", 39}
  RD: [4] "b(i,j)" = 616.534385500486; {"test.f", 39}
  AW: [4] "a(i,j)" = 616.534385500486; {"test.f", 39}
  <...>
```

Временные затраты и размеры трасс

Сетка	10x10 100 итераций		100x100 100 итераций		1000x1000 1000 итераций	
	Размер трассы	*Время работы*	**Размер трассы**	*Время работы*	**Размер трассы**	*Время работы*
Без отладки	**-**	*0.42 s*	**-**	*0.45 s*	**-**	*3.02 s*
Отладка без оптимизаций	**4.6 Mb**	*0.5 s*	**694 Mb**	*9.35 s*	**~800 Gb**	*~12000 s*
Контрольные суммы	**24 Kb**	*0.46 s*	**24 Kb**	*6.0 s*	**239 Kb**	*~7000 s*
Граничные итерации	**387 Kb**	*0.45 s*	**390 Kb**	*0.47 s*	**3.89 Mb**	*3.34 s*

Проблемы текущей реализации отладки

- **Ресурсы**, требуемые для полной отладки программы, делают практически невозможным работу с большими входными данными
- Оптимизации отладки дают больше возможностей, но не позволяют точно локализовать момент возникновения ошибки
  - Контрольные суммы массивов позволяют определить итерацию, на которой произошло расхождение и массивы, значения в которых были вычислены неверно
  - Метод граничных итераций во многих случаях способен лишь примерно обозначить когда произошло расхождение

Проблемы текущей реализации отладки

- **Точность вещественных вычислений**: любая вещественная операция является потенциальным источником ошибок
- Результаты вещественных операций умножения и деления могут не совпасть в 1-2 младших знаках мантиссы при запуске:
  - на разных машинах
  - используя различные компиляторы
  - используя различные опции одного компилятора
- Изначально расхождения игнорируются отладчиком, но они:
  - накапливаются, вплоть до существенных для отладки значений
  - распространяются на другие переменные, участвующие в вычислениях
  - могут привести к расхождению потока управления двух программ

Развитие метода сравнительной отладки

- **Реализация второго основного подхода к отладке**: обмена отладочной информацией между двумя параллельно работающими программами

- Запуск двух инструментированных для отладки программ, синхронизация и обмен блоками трассы по сети
  - Вариант для кластеров – запуск двух вариантов программы в рамках одного пространства процессов *(например коммуникатор MPI)*
- Стратегия разбиения трассы на блоки:
  - Границы блоков – начала и концы параллельных циклов
  - Промежуточные результаты программы в данных точках всегда должны быть идентичными, вне зависимости от количества процессов и конфигурации вычислительной системы

Развитие метода сравнительной отладки

- **Реализация второго основного подхода к отладке**: обмена отладочной информацией между двумя параллельно работающими программами

- Время работы отладчика увеличится:
  - Необходимо синхронизировать работу параллельно запущенных программ
  - Продолжение вычислений возможно только после обработки полученного блока трассы

- Затраты на дисковую память практически нулевые
  - Необходимо лишь хранить в оперативной памяти отладчика текущий блок трассы, ожидающий следующего этапа проверки

Развитие метода сравнительной отладки

- **Режим коррекции вещественных вычислений**: замена результата вещественной операции на соответствующее значение из трассы
- Текущая реализация отладчика поддерживает похожий механизм для коррекции редукционных переменных
- Исключает применение оптимизаций размера файла трассы – необходима трассировка всех вещественных операций
- Совместим с режимом параллельной работы программ
  - Отладка начинается только после получения текущего блока трассы от второй программы

Спасибо за внимание

DVM-**СИСТЕМА**

http://dvm-system.org

dvm@keldysh.ru