LuNA-2: Результаты за 2018 год

Докладчик: Беляев Н. А., аспирант 2 курса ИВМиМГ СО РАН
Научный руководитель: д. т. н., проф., Малышкин В. Э.

Введение

- Решение задач численного моделирования с использованием суперкомпьютеров требует разработки параллельных программ
- Разработка параллельных программ для решения задач численного моделирования зачастую требует от разработчика навыков системного параллельного программирования

Введение

- Системное параллельное программирование не является частью предметной области
- Абстрагирование разработчика параллельной программы для решения задач численного моделирования от решения задач системного программирования позволит сократит время разработки параллельных программ

Требования к системам параллельного программирования

- Абстрагирование разработчика параллельной программы от зада системного параллельного программирования
- Обеспечение уровня производительности параллельных сравнимого с уровнем, достигаемым при ручной реализации программ

Существующие средства и системы ПП

- MPI
- OpenMP
- DVM-H, sapfor
- Charm++
- OpenCL

Система LuNA

Общие сведения

- Система LuNA – это система автоматического конструирования параллельных программ, ориентированная на решение задач численного моделирования на суперкомпьютерах

Параллельная программа в системе LuNA

- ПП в системе LuNA представляется я виде множества фрагментов вычисления (ФВ) и фрагментов данных (ФД)
- Также ПП в системе LuNA может быт представлена в виде двудольного ориентированного графа, вершинами которого являются ФВ и ФД, а дуги обозначают информационные зависимости

Термины и определения

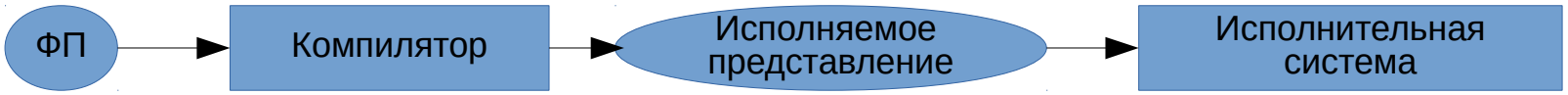
- Управление (на множестве ФВ) - отношение частичного порядка (на множестве ФВ), накладывающее ограничения на порядок исполнения ФВ

Исполнение фрагментированной программы

- Исполнение ФВ в системе LuNA происходит по т. н. Базовому алгоритму, ФВ исполняется по готовности значений всех своих входных ФД
- В языке LuNA присутствуют ФВ двух видов:
  - Атомарные ФВ, представляющие собой вызовы процедур языка Си
  - Структурированные ФВ (if, for, while, sub) – представляют собой ФВ, исполнение которых заключается в поступлении на исполнение некоторого множества ФВ, называемого телом структурированного ФВ. Исполнение структурированного ФВ в дальнейшем условно будем называть “раскруткой” ФВ

Архитектура системы LuNA

- Система состоит из компилятора языка LuNA и исполнительной системы



Недостатки (проблемы) системы LuNA

- Неконтролируемая “раскрутка” структурированных ФВ во время исполнения ФП
- Неэффективные алгоритмы распределения фрагментов по узлам мультикомпьютера.
- Проблема «сборки мусора»
- Использование универсальных алгоритмов динамической балансировки вычислительной нагрузки на узлы мультикомпьютера
- Решения о выборе управления на подмножествах ФВ принимается динамически, что снижает производительность системы

Система LuNA-2

LuNA-2

- Система LuNA-2 должна автоматически конструировать параллельные программы, обладающие свойством настраиваться на конкретный вычислитель (в т. ч., свойством обеспечения при необходимости динамической балансировки вычислительной нагрузки) для заданного класса численных алгоритмов
- Производительность сконструированных программ не должна быть хуже таковой у программ, разработанных “вручную” более чем в 2 раза

Цель работы

- Разработка системных алгоритмов эффективного исполнения фрагментированных программ на мультикомпьютере для заданного класса задач численного моделирования и реализация алгоритмов в виде средства параллельного программирования рамках системы LuNA

Задачи

Разработать алгоритмы распределения фрагментов данных и фрагментов вычислений на узлы мультикомпьютера

- Разработать алгоритмы анализа ФА для выбора походящего алгоритма распределения фрагментов на узлы мультикомпьютера
- Разработать алгоритм анализа результатов измерения производительности системы во время предыдущих запусков фрагментированной программы (ФП) (т. н. профиля исполнения ФП)
- Разработать алгоритмы динамического перераспределения фрагментов на узлы мультикомпьютера (алгоритмы динамической балансировки вычислительной нагрузки)

Задачи

- Разработать алгоритмы выбора алгоритма динамической балансировки для множеств фрагментов, при исполнении которых наблюдается дисбаланс вычислительной нагрузки
- Разработка алгоритмов своевременного освобождения памяти, занимаемой значениями ФД, уже потребленными всеми ФВ, для которых данное значение было входным (алгоритмы «сборки мусора»)
- Разработка алгоритмов выбора управления, согласно которому будет осуществлено исполнение фрагментов на узлах мультикомпьютера.
- При выборе управление алгоритмы должны по возможности сокращать общее время исполнения ФП и потребление ресурсов мультикомпьютера во время исполнения. В частности, необходимо, по возможности, выбрать управление, позволяющее максимально быстро освобождать память, занимаемую значениями ФД

Детали реализации системы LuNA-2

Общие сведения

- Система LuNA-2 на текущий момент представляет собой интеллектуальный компилятор
- Компилятор выделяет из ФА множество фрагментов, для которых возможна генерация статического управления и распределения ресурсов и осуществляет генерацию в случае, если это возможно

Общие сведения

- Подмножества фрагментов, для которых генерация статического управления и распределения ресурсов, исполняются системой LuNA
- Для облегчения анализа ФА и повышения уровня программирования язык LuNA был дополнен

Изменения языка LuNA

- Ключевое слово reduction, пример:<reduce max>(D,diff, real, cf_reduce(%in, %out));
- Ключевое слово borders_exchange, пример:<borders_exchange 1>(Fi(it), B, read(%in, %1, %2), write(%out, %1, %2, %FRAG_IDX));
- Сущность «массив ФД», пример: DFArray A[n][n];
- Явное выделение итерационного процесса над множеством массивов ФД в описании ФА:
  while (it < 500), it = 0 .. out iter : <Fi(it) --> Fi(it+1)>{…}
- Блок dynamic: указание множества ФВ, вносящих дисбаланс нагрузки

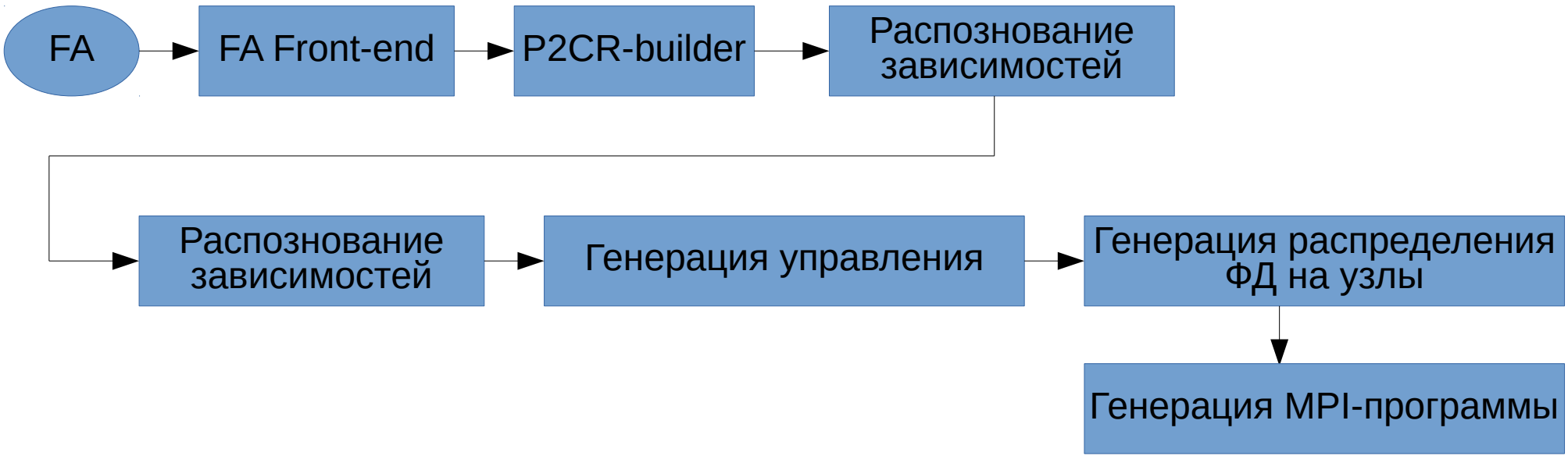
Текущие ограничения, накладываемые на ФА

- Все массивы ФД в программе одномерные или двумерные
- Доступ к элементам массивов ФД имеет вид A[df1][df2]...[dfN]

Требования к архитектуре компилятора LuNA-2

- Модульность
- Универсальность внутреннего представления ФА
- Универсальность модулей (каждый модуль должен содержать реализацию преобразования над универсальным внутренним представлением и может быть применён к нему на любой стадии компиляции)

Процесс компиляции LuNA-2-программы



Требования к внутреннему представлению

- Универсальность
- Возможность представления информации о семантической структуре ФА
- Возможность представления информации о подмножествах ФВ, для которых возможно параллельное исполнение и требуется распределение на узлы мультикомпьютера

Внутреннее представление P2CR (основные определения)

- P2CR (Phase 2 Common Representation)- основная структура данных фазы 2
- P2CR - Ориентированный граф (возможны циклы)
- Область — совокупность непустого множества массивов ФД, имеющих одинаковую размерность и одинаковые размеры по каждому измерению и множества ФД, значения которых задают размерности массивов
- Блок — множество ФВ
- Вершины соответствуют ФД, т. н. областям и т. н. блокам, а ребра — информационным зависимостям между блоками и областями
- Дуга от области (блока) к блоку (области) означает, что ФВ блока потребляют (вырабатывают) значения ФД или значения элементов массивов ФД области согласно некоторому множеству выражений языка LuNA (эти выражения являются «меткой» каждой дуги графа)
- Дуга от ФД (блока) к блоку (ФД) имеет аналогичный смысл

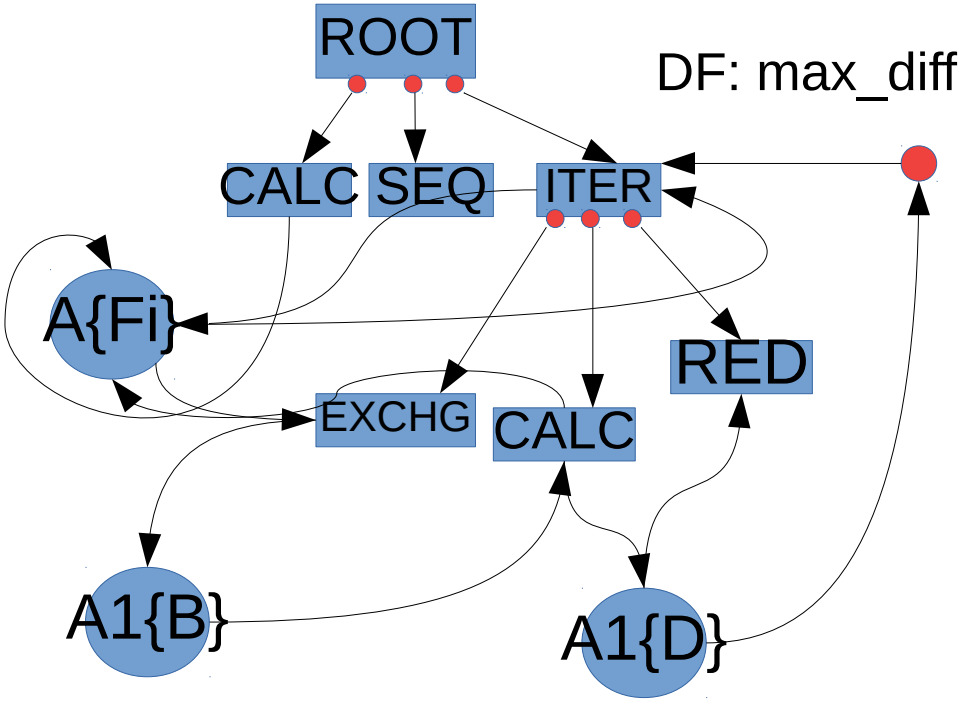
Внутреннее представление P2CR (основные определения)

- Дуга от блока к блоку задает либо прямое управление между блоками, либо указывает на то, что ФВ блока поступают на исполнение вследствие исполнения структурированного ФВ, содержащегося в блоке-начальной вершине дуги, в зависимости от метки, присвоенной дуге

Внутреннее представление P2CR

- Блоки имеют тип аналогично типу ФВ блока:
  - ITER — блок, содержащий ФВ, задающий итерационный процесс на расчётной области (содержит ФВ типа for или while)
  - CALC — блок, содержащий множество ФВ, реализующих непосредственно вычисления на расчётной области
  - SEQ — блок, содержащий ФВ, которые необходимо выполнить последовательно на одном или нескольких из узлах мультикомпьютере
  - RED — блок, содержащий ФВ, реализующие редукцию на массиве ФД
  - EXCHG — блок, содержащий ФВ, реализующие обмен теневыми гранями между ФД

Внутреннее представление P2CR




```
sub main()
{
  df iter, max_diff(2);
  DFArray Fi[500];
  for i=0..499 {
    init_fi(Fi(0)[i], i);
  }
  while (it < 500 ), it = 0 .. out iter : <Fi(it) --> Fi(it+1)>
  {
    df diff;
    DFArray B[500], D[500];
    <borders_exchange 1>(Fi(it), B, read(%in, %1, %
        write(%out, %1, %2, %FRAG_IDX));
    assign(max_diff(it+1), diff);
    for i = 0 .. 499 {
      cf clc[i] : poi(B[i], i, Fi(it+1)[i], D[i]);
    }
    <reduce max>(D,diff, real, cf_reduce(%in,  %out)
  }
init_max(max_diff(0));
}
```

AAPIR

- AAPIR (Arrays Access Patterns Intermediate Representation) — это внутреннее представление для хранения информации о распознанном компилятором характере информационных зависимостей в программе

AAPIR

- AAPIR состоит блоков, прикрепляемых компилятором к каждому узлу графа P2CR
- Блок состоит из множества описателей характера доступа к некоторому массиву ФД
- Один описатель — это пара <T, <In, Out>>, T- тип доступа (последовательный или без зависимости между элементами массива) ,

  In — множество входных массивов ФД, Out- множество выходных массивов ФД

Домены распределения нагрузки

- Доменом будем называть упорядоченную пару <A, I>, где A = {Array1...ArrayN} — множество массивов ФД, имеющих одинаковую размерность, I={{i11,….i1N}, {i21, …},…} - множество возможных индексов
- Размерностью домена будем называть размерность входящих в него массивов ФД

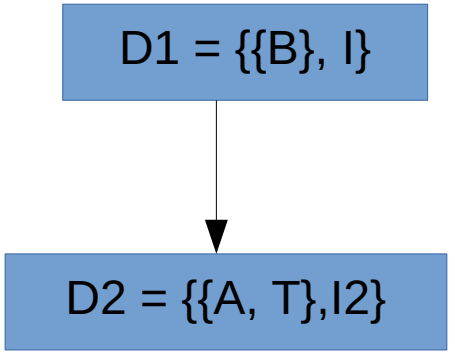
Дерево распределения ФД

- Для генерации распределения ФД на узлы мультикомпьютера компилятор строит дерево доменов
- Узлами дерева являются обнаруженные (на основе анализа AAPIR) компилятором домены массивов ФД
- При генерации программы, сначала генерируется (в случае необходимости) вызов процедуры балансировки выполняется для массивов, входящих в корень дерева. Остальные домены изменяются в ходе балансировки согласно алгоритму балансировки.

Дерево распределения ФД

```
DFArray A[n][n];
…
While  (…), it=0..out iter : <A(it) → A(it+1)>
{
DFArray T[n][n];
for i = 0...n-1
{
        DF B[n];
        for j = 0..n-1
        {
                B[i] = cf(A[i][j]);
                dynamic {
                // some calculations with B
                }
        }
}

}
```



Внутреннее представление FPASM

- FPASM (Fragmented Program Assembler) — представляет собой множество команд, на котором задано отношение частичного порядка (управление), определяющее порядок исполнения команд

Внутреннее представление FPASM

- Краткий список команд FPASM:
  - Исполнение ФВ
  - Исполнение множества ФВ
  - Редукция
  - Обмен теневых граней
  - Акт балансировки нагрузки

Динамическая балансировка нагрузки

Основные сведения

- Алгоритм балансировки вычислительной нагрузки реализован в библиотеки времени выполнения, используемой сгенерированной программой
- Пользователь испольщует оператор dynamic для выделения множества ФВ, которые вносят основной дисбаланс нагрузки

Основные сведения

- Компилятор LuNA-2 генерирует MPI-программу, содержащую вызовы в соответствующих местах реализованного в библиотеке балансировщика
- В перспективе возможен выбор компилятором алгоритма балансировки из нескольких доступных в библиотеке, а также, подбор параметров алгоритма на основании результатов профилирования

Алгоритм динамической балансировки

- Используется синхронный алгоритм
- Нагрузкой на узел мультикомпьютера считается время, за которое выполняется блок dynamic

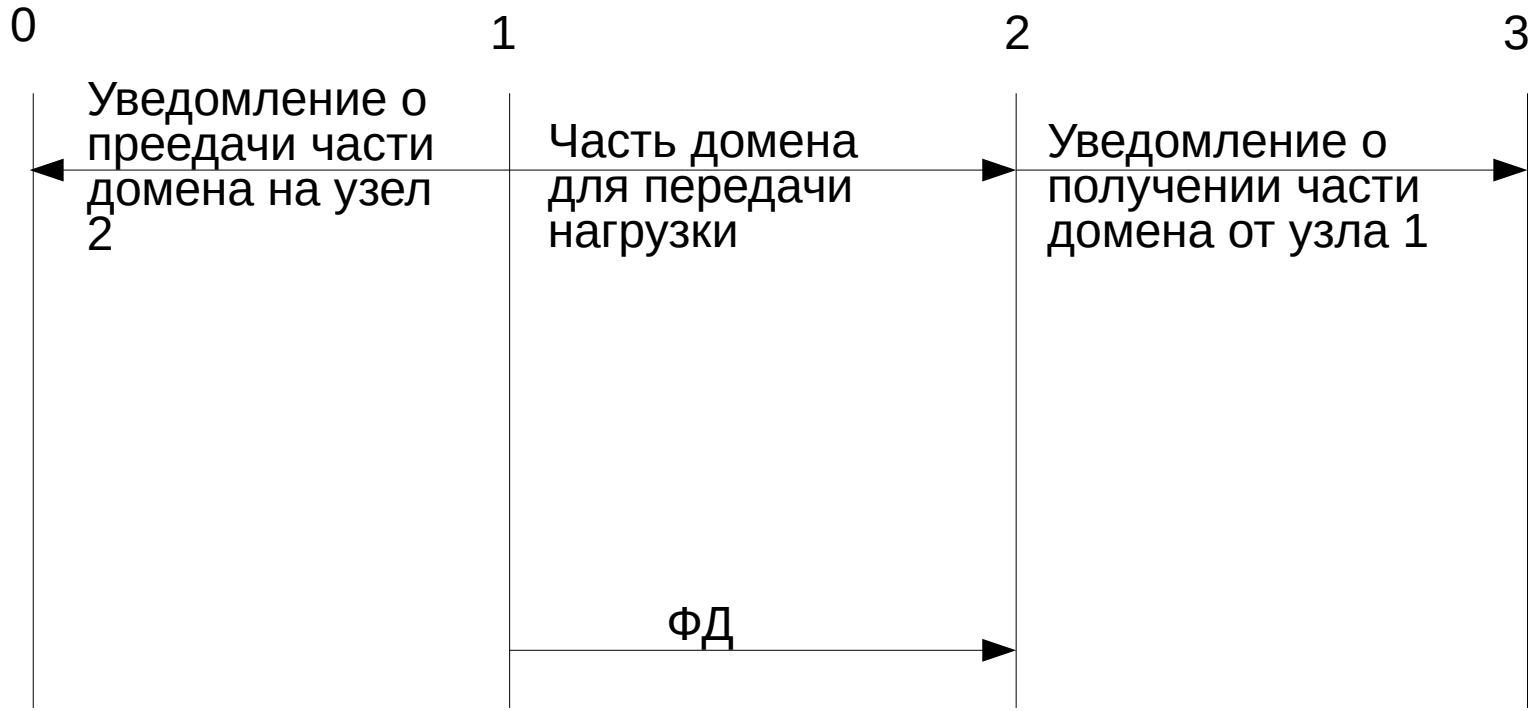
Алгоритм динамической балансировки нагрузки

- Будем называть 2 ФД из некоторого массива ФД размерности N соседями, если их индексы по 0 < n <= N-1 измерениям отличаются на единицу (в рамках текущего поддерживаемого класса программ)
- Будем называть узлы мультикомпьютера соседними, если на них хранятся соседние ФД из некоторых массивов

Алгоритм динамической балансировки нагрузки

1. Вычисление нагрузки на текущем узле (время выполнения блока dynamic)
2. Обмен нагрузками с соседними с текущим узлами (после этого шага на каждом узле известны нагрузки соседних узлов)
3. Выбор соседнего узла для отправки ФД. Выбирается узел, разность нагрузки с которым максимальна
4. Вычисление количества ФД для отправки. Отправляются ФД с наиболее нагруженного узла на менее нагруженный
5. Отправка части домена на узел и обновление информации о соседних ФД в домене
6. Отправка непосредственно ФД

Обновление информации о соседних ФД в домене



Параметры алгоритма

- Интервал между актами балансировки
- Функция, вычисляющая количество передаваемых ФД от разницы нагрузки и количества ФД, доступных на узле
- Минимально необходимый дисбаланс нагрузки между соседними узлами для инициирования балансировки

Результаты сравнительных тестов производительности

Тест 1

- Оборудование: 16 узлов x 16 ядер (256 MPI-процессов) кластер mvs10k
- Задача: PIC-3D, 2D-фрагментация, 400000000 частиц, 1000 шагов по времени, 160x160 ФД, размер сетки: 160x160x100, R=0.3 (радиус диска),

  координаты диска в расчётной области: (1.0, 1.0, 0.5), размер области: 2.0x2.0x1.0
- Время работы без балансировки: 6417 sec
- Время работы с балансировкой: 5195 sec (ускорение в ~1.2 раза)

Тест 2

- Оборудование: 16 узлов x 16 ядер (256 MPI-процессов) кластер mvs10k
- Задача: PIC-3D, 2D-фрагментация, 400000000 частиц, 1000 шагов по времени, 160x160 ФД, размер сетки: 160x160x160, R=0.5 (радиус диска),

  координаты диска в расчётной области: (1.0, 1.0, 1.0), размер области: 2.0x2.0x1.0
- Время работы без балансировки: 3267.31 sec
- Время работы с балансировкой: 3006.31 sec (ускорение на 8%)

Спасибо за внимание!