1. Аннотация

В данной работе исследованы способы генерирования параллельных программ с заданным управлением, используя систему LuNA. Целью работы была разработка языковых средств для описания параллельных программ, создание структур данных и алгоритмов для их конструирования, а также экспериментальное исследование эффективности. Автоматическое конструирование параллельных программ без использования исполнительной системы имеет преимущество в экономии ресурсов. Однако такой подход требует специфических решений для разных языков программирования. В ходе работы был проведен анализ существующих решений и сформулированы задачи для достижения цели. Результатом была разработка подсистемы прямого управления исполнением фрагментированных программ, которая может быть использована как генератор параллельных фрагментированных программ.

2. Тема работы

Разработка подсистемы прямого управления исполнением фрагментированных программ в системе LuNA.

3. Состав коллектива

Студент: Пирожков Андрей Константинович, Факультет информационных технологий, кафедра параллельных вычислений, 4-й курс, группа 19201.

Научный руководитель: Маркова Валентина Петровна, к.т.н., доцент. параллельных вычислений ФИТ НГУ, с.н.с. ИВМиМГ СО РАН.

Соруководитель: Перепёлкин Владислав Александрович, ст. преп. каф. параллельных вычислений ФИТ НГУ, н.с. ИВМиМГ СО РАН.

4. Научное содержание работ

4.1.Постановка задачи

Для достижения поставленной цели, необходимо разработать входное представление, которое опишет фрагментированную программу, состоящую из множества фрагментов данных и фрагментов вычислений, которые пойдут на вход генератору параллельных программ. Для упрощения постановки задачи в генераторе параллельных фрагментированных программ примем ограничение,

что количество фрагментов вычислений и фрагментов данных должно быть заранее известно. Тогда в генераторе будут использоваться только атомарные операторы. Для описания входного представления будут разработаны языковые средства описания фрагментированной программы, которые будут описывать множество фрагментов данных и вычислений. Для реализации прямого управления, нужно разработать конструкции, из которых будет конструироваться параллельная программа. Такие конструкции будут реализовывать часть фрагментированной программы, отображенного на конкретный ресурс и осуществлять конкретное управление. А алгоритм конструирования должен будет построить из конструкций параллельную программу в порядке информационных зависимостей, а затем сгенерировать параллельный код, который будет выполняться параллельно на нескольких процессах. Также необходимо рассмотреть вопрос эффективности генерируемых параллельных программ.

4.2.Современное состояние проблемы

Существует 2 основных подхода решения проблемы автоматического конструирования параллельных программ: создание параллельной программы из существующей последовательной и использование систем автоматического конструирования параллельных программ.

Первый подход заключается в добавлении конструкций в исходный код программы для создания параллельной реализации. Чаще всего они распараллеливают циклы, в которых обрабатываются массивы. Также чаще всего инструменты предназначены, чтобы работать в общей памяти а не распределенной. А такие программы не очень хорошо подходят для суперкомпьютеров с распределенной памятью.

А при втором подходе, системы автоматического конструирования параллельных программ используют исполнительную систему.

Также были рассмотрены технологии с помощью, которых работают все вышеперечисленные инструменты.

4.3.Подробное описание работы, включая используемые алгоритмы

Цель и задачи

Цель работы: разработать подсистему автоматического конструирования из LuNA-программ традиционных императивных параллельных программ без использования исполнительной системы.

На вход подсистемы идёт LuNA-программа, а на выходе получается традиционная императивная параллельная программа без исполнительной системы. Соответственно, цель работы: разработка генератора, который осуществит данное преобразование.

Цель достигается выполнением следующих задач:

- Разработка языковых средств описания фрагментированной программы.
- Разработка структур данных и алгоритмов, обеспечивающих конструирование параллельных программ.
- Экспериментальное исследование эффективности конструируемых параллельных программ.

Фрагментированное программирование

Введем необходимые термины:

Во фрагментированной LuNA-программе вычисления рассматриваются как совокупность фрагментов данных и вычислений.

Фрагмент данных — это блок данных, который используется во фрагментированной программе как входной или выходной параметр для фрагмента вычислений

Фрагмент вычислений — это объект времени исполнения, который описан в фрагментированной программе, который содержит описание входных и выходных фрагментов данных. Фрагменты вычислений выполняются по готовности данных.

Чтобы достичь поставленных целей и задач было произведено упрощение в понятии фрагментированной программы. И в рамках работы фрагментированная программа будет состоять из конечного количества объектов, количество которых заранее известно.

Предлагаемая структура языка:

Входное представление фрагментированной программы было описано с помощью входного языка. Были описаны основные элементы фрагментированной программы:

- Входные и выходные фрагменты данных, количество используемых процессов.
- Перечисление фрагментов вычислений с подробным их описанием.
- Перечисление всех фрагментов данных и переменных с указанием типов.
- Указание переменных по умолчанию (define).

Структура описания входного языка предоставляет достаточную информацию для того, чтобы из неё можно было конструировать параллельную фрагментированную программу.

Предлагаемые конструкции

Входное представление в генераторе параллельных программ преобразуется в конструкции. Было разработано несколько конструкций с помощью которых алгоритм будет строить параллельную фрагментированную программу.

У каждой конструкции есть заготовки кода, которые будут применяться с нужными параметрами при генерации кода.

- ProgSpec – конструкция с описанием множества фрагментов вычислений и фрагментов данных. Такая конструкция хранит в себе данные описанной фрагментированной программы, которые используют другие конструкции.
- Mcell – конструкция ячейки памяти, описывающая используемые переменные и фрагменты данных.
- Port – вспомогательная конструкция, которая описывает аргументы, применяемые фрагментом вычислений. Конструкция port является частью конструкции Sniplet

- Sniplet – это конструкция, предназначенная для выполнения операций. Такая конструкция имеет несколько видов, у которых разные назначения.
  - ArgvSniplet – сниплет ввода из переданных аргументов в программе
  - VarToStdoutSnip – сниплет вывода в консоль
  - InvokeOperationSnip – сниплет подстановки операции. Именна эта конструкция чаще всего используется генератором. Она отвечает за генерацию кода для фрагментов вычислений.
  - SendRecvSniplet – коммуникационный сниплет. Используется для создания коммуникаций между разными процессами. Генерирует код и осуществляет передачу информации с помощью коммуникационного интерфейса MPI.
- PartialProgram – это основная конструкция, в которой формируется частично-определенная программа. Эта конструкция содержит в себе все вышеперечисленные конструкции.

Предложенные конструкции позволяют выразить отображение фрагментов вычислений на разные процессы, а также осуществлять прямое управление фрагментированной программы. Этих средств достаточно для конструирования некоторого класса практических задач.

Алгоритм конструирования программы

С помощью предложенных конструкций, был предложен алгоритм конструирования программы строит паралелльную программу в несколько этапов:

На первом этапе, параллельная программа конструируется без коммуникаций, причем итерационно, пока не сработает критерий завершенности конструируемой программы. При этом осуществляет 2 шага:

- На 1 шаге создаются необходимые конструкции ячеек памяти, если они еще не созданы, а затем в сниплетах порты строят связи с ячейками памяти.

- А на 2ом шаге – если существует неинициализированная ячейка памяти, то создаётся сниплет согласно входному представлению программы, в которой эта ячейка памяти инициализируется.

На втором этапе осуществляется добавление коммуникаций в построенной программе. Т.е. в частично-определённую программу добавляются коммуникационные сниплеты.

На третьем этапе происходит генерация кода сконструированной параллельной программы. После этого этапа пользователь получает готовый параллельный код.

Заключение

В результате работы была разработана подсистема прямого управления исполнением фрагментированных программ. В ходе работы были выполнены все поставленные задачи тем самым была достигнута цель работы.

4.4.Полученные результаты

Во время тестирования проверялось требование, чтобы любой сгенерированный код программы должен быть компилируемым и выполнимым без ошибок при верном формате входного представления.

Основной задачей тестирования на простых примерах является – это проверка конструирования параллельной программы для разного количество процессов, а также проверка работы всех разработанных конструкций. Все простые тесты прошли успешно.

В качестве практического значимого примера для тестирования, было выбрано решение уравнения Пуассона в трехмерной области методом Якоби с одномерной декомпозицией. Его тестирование производилось на вычислительном кластере НГУ. Результаты тестирования на 4 узлах по 8 процессов каждый (всего 32 MPI-процесса) получились следующие:

- Сгенерированная программа с неэффективным распределением (это обычный среднестатистический сгенерированный генератором вариант параллельной программы) — 57.158 с.

- Сгенерированная программа с эффективным распределением (это где пользователь самостоятельно указывает во входном языке на каком процессе исполнять фрагмент вычислений) — 38.854 с.
- LuNA-программа эффективным распределением по узлам (это LuNA реализация примера, в котором задействованы основные способы повышения эффективности программы) — 44.892 с.

Благодаря прямому управлению удалось достичь более лучшего результата выполнения сгенерированной программы, чем при выполнении LuNA-программы. Разница составила 6 секунд. Это примерно на 13% быстрее. Это также косвенно подтверждает, что в системе LuNA часть ресурсов тратится на исполнительную систему, а разработанному генератору она не требуется.

5. **Эффект от использования кластера в достижении цели работы**

Кластер ИВЦ НГУ позволил проверить работоспособность сгенерированных параллельных фрагментированных программ на большом количестве процессов. Также кластер позволил сравнить эффективность сгенерированной программы с LuNA-программой на 32 MPI-процессах.

6. **Перечень публикаций содержащие результаты работы.**

Пирожков А. К. Разработка подсистемы прямого управления исполнением фрагментированных программ в системе LuNA / Пирожков А. К. //Информационные технологии : Материалы 61-й Междунар. науч. студ. конф. 17–26 апреля 2023 г. / Новосиб. гос. ун-т. — Новосибирск : ИПЦ НГУ, 2023 (Принято в печать).