Разработка и реализация алгоритмов вызова распределённых MPI-подпрограмм из LuNA-программ

Автор: Изотов А.С.

Актуальность проблемы

- С ростом количества задач, которые требуют больших объёмов вычислений, появились системы параллельного программирования (Charm++, Legion).
- Программы, созданные с помощью таких систем, проще в разработке, но, зачастую проигрывают в быстродействии программам написанным с использованием стандартных средств параллельного программирования (MPI).
- Предлагается реализовать в системе LuNA возможность вызова MPI-подпрограмм, чтобы использовать высокую производительность MPI, где это критично.

Цели и задачи

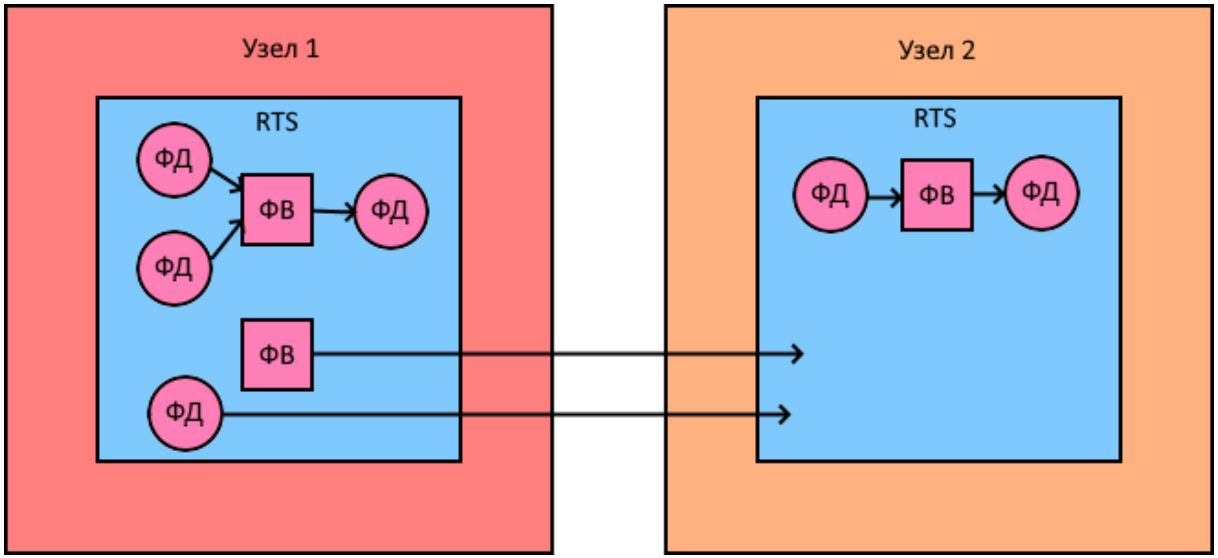
- **Цель работы** – Разработать и реализовать подсистему вызова MPI-подпрограмм из LuNA-программ.
- Задачи:
  - Анализ систем параллельного программирования и их возможностей по запуску подобных распределённых подпрограмм.
  - Разработка интерфейса для вызова распределённых MPI-подпрограмм из LuNA-программ.
  - Разработка необходимых алгоритмов для обеспечения работы интерфейса.
  - Программная реализация интерфейса и алгоритмов.
  - Экспериментальное исследование работы программной реализации.

Обзор систем параллельного программирования

- Был проведён обзор систем параллельного программирования и их возможностей по запуску распределённых программ, таких как:
  - Charm++
  - Legion
  - HPX
  - LibGeoDecomp
- Во всех вышеперечисленных системах существуют разные реализации вызовов распределённых подпрограмм, то есть не существует универсального подхода, и у каждой системы свой индивидуальный подход.

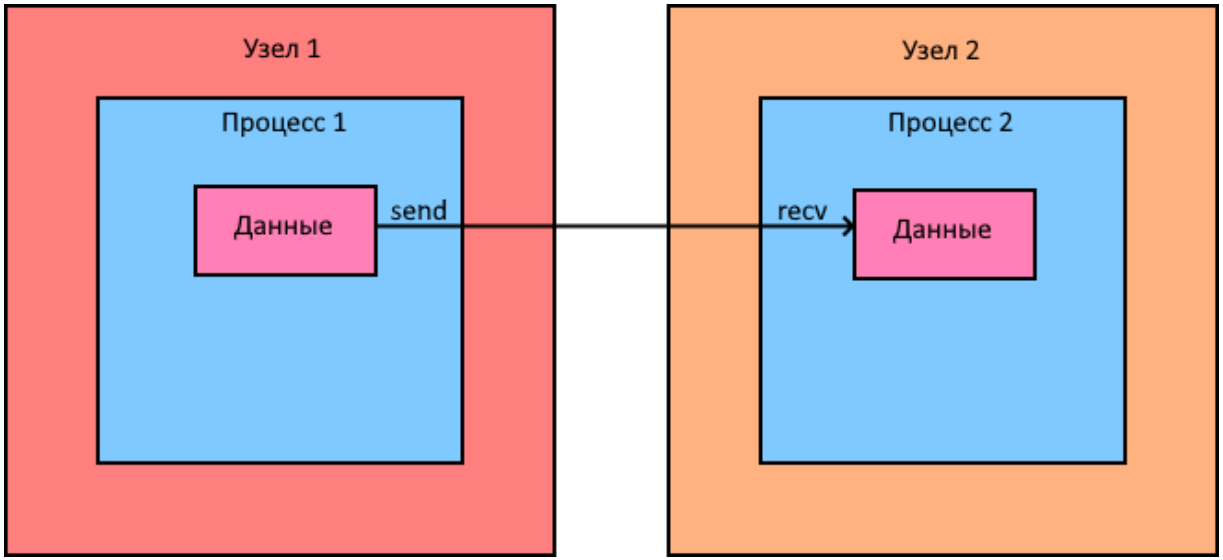
Система LuNA

- Система фрагментированного программирования LuNA – задачеориентированная система параллельного программирования.
- Задачей в системе LuNA является фрагмент вычислений. Фрагменты вычислений потребляют и вырабатывают фрагменты данных – конечные совокупности переменных.
- Исполнение программы в системе LuNA производится с помощью runtime-системы (системы исполнения).
- Runtime-система пересылает различные задачи на другие узлы, за счёт чего достигается параллелизм.
- Фрагменты вычислений и фрагменты данных перемещаются между вычислительными узлами асинхронно.



Message Passing Interface

- Программы написанные с использованием стандарта MPI реализуют концепцию процедурного программирования.
- MPI-программы – это несколько последовательных процессов, запущенных на разных вычислительных узлах, которые взаимодействуют друг с другом с помощью сообщений.



Постановка задачи

- Требуется разработать интерфейс обеспечивающий возможность вызова MPI-подпрограмм из LuNA-программ.
- Основной проблемой при разработке интерфейса для взаимодействия системы исполнения LuNA и MPI-подпрограммы, является трудносовместимость двух разных моделей вычисления.

Требования к интерфейсу

- Необходимо установить требования, которым данный интерфейс должен удовлетворять:
  - MPI-подпрограмма должна реагировать на появление фрагментов данных.
  - Быть достаточно эффективным, т.е. сам интерфейс не должен навязывать какие-либо ресурсоёмкие процессы.
  - Иметь возможность исполнять распределённую подпрограмму, которая может использовать средства и функции MPI.
  - Быть способным получать и передавать обратно фрагменты данных.
  - Быть способным исполнять подпрограмму на нескольких узлах.

Описание разработанного интерфейса

- Следующие интерфейсные методы определяются в MPI-подпрограмме
  - virtual void request_dfs()
  - virtual void on_receive_df(int, const DF&)
- Основными частями алгоритма взаимодействия runtime-системы и MPI-подпрограммы через разработанный интерфейс являются:
  1. Вызов MPI-подпрограммы из программы на языке LuNA.
  2. Передача информации о фрагментах данных, используемых в подпрограмме.
  3. Запрос необходимых фрагментов данных на каждом узле.
  4. Обработка фрагментов данных.
  5. Передача результата на определённый узел.

Программная реализация

- Интерфейс был реализован в виде нового класса SubroutineInstance в исходном коде системы фрагментированного программирования LuNA.
- Также поддержка вызова этого интерфейса была реализована в трансляторе и runtime-системе.
- Был расширен синтаксис языка LuNA:
  - Объявление MPI-подпрограммы:

    import <lib_func_name> (<params>) as <code_id>: MPI ;
  - Вызов MPI-подпрограммы:

    code_id(<args>) @ { number_dfs: n; … };

Тестирование

- Для проверки работоспособности и быстродействия разработанного решения была разработана LuNA-программа умножения матриц, и её часть – редукция, реализована в двух вариантах: в виде LuNA-подпрограммы и MPI-подпрограммы.
- Желательным результатом является уменьшение времени выполнения LuNA-программы с MPI-подпрограммой.

Тестирование

- Размер матрицы 32x32, размер фрагментов данных 100x100, 1 рабочий поток.

	Время выполнения (количество узлов), с.			
Наличие MPI-подпрограммы	1	2	4	8
Нет	79,73	111,41	15,86	18,85
Да	39,76	128,10	27,95	25,34

- Размер матрицы 16x16, размер фрагментов данных 200x200, 1 рабочий поток.

	Время выполнения (количество узлов), с.			
Наличие MPI-подпрограммы	1	2	4	8
Нет	83,78	55,16	12,97	8,02
Да	43,02	44,72	7,88	8,77

Тестирование

- Размер матрицы 32x32, размер фрагментов данных 100x100, 12 рабочих потоков.

	Время выполнения (количество узлов), с.	
Наличие MPI-подпрограммы	1	2
Нет	45,81	51,28
Да	41,78	43,74

- Размер матрицы 16x16, размер фрагментов данных 200x200, 12 рабочих потоков.

	Время выполнения (количество узлов), с.	
Наличие MPI-подпрограммы	1	2
Нет	45,88	51,44
Да	42,62	44,63

Тестирование. Выводы

- В первую очередь, результаты тестирования свидетельствуют о том, что разработанное решение для вызова MPI-подпрограмм работает правильно.
- Также результаты тестирования показывают что, несмотря на увеличение накладных расходов, связанных с применением разработанного интерфейса, для определённого класса задач, выигрыш в производительности MPI-подпрограмм перевешивает эти накладные расходы. Таким образом, данный интерфейс может быть использован для повышения производительности LuNA-программы.

Заключение

- В результате работы была разработана и реализована подсистема вызова MPI-подпрограмм из LuNA-программ, в частности:
  - Разработан интерфейс для вызова распределённой MPI-подпрограммы из LuNA-программы.
  - Разработан алгоритм вызова распределённой MPI-подпрограммы из LuNA-программ.
  - Разработанный интерфейс и алгоритм были реализованы и встроены в систему LuNA.
  - Проведено тестирование, которое подтверждает работоспособность и эффективность предложенного решения.

Направления дальнейшей работы

- Избавление от обязательного указания количества входных фрагментов данных при вызове MPI-подпрограммы.
- Доработка интерфейса, которая подразумевает возможность использования ограниченного числа узлов, для выполнения MPI-подпрограммы.
- Добавление альтернативного интерфейса, который будет более понятным и привычным с точки зрения MPI-программ.

Разработка и реализация алгоритмов вызова распределённых MPI-подпрограмм из LuNA-программ

Автор: Изотов А.С.