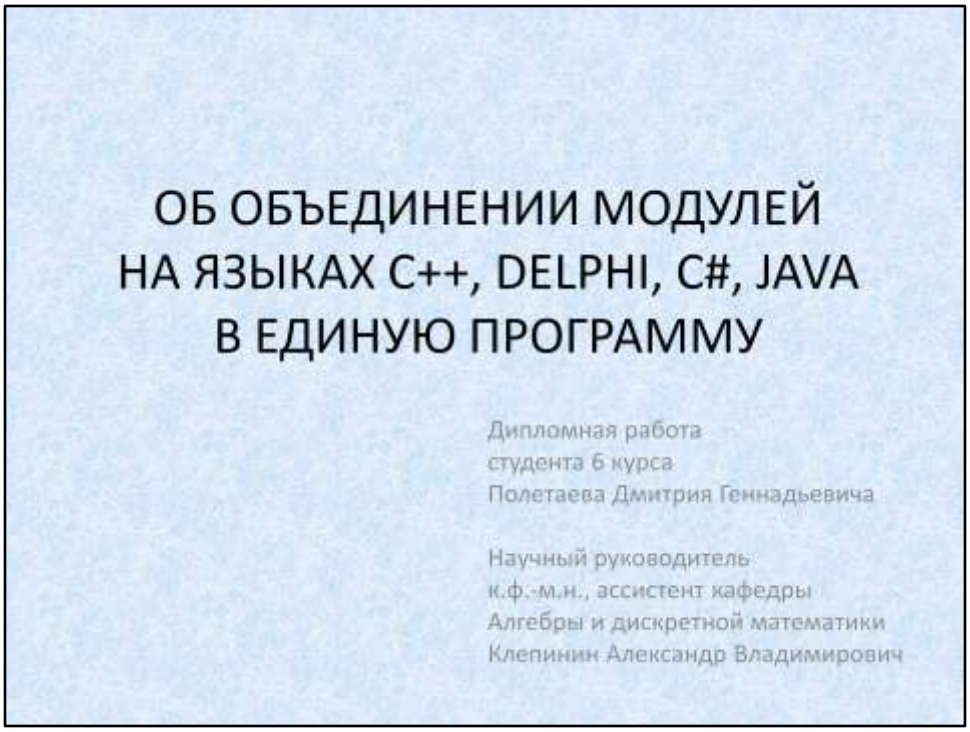
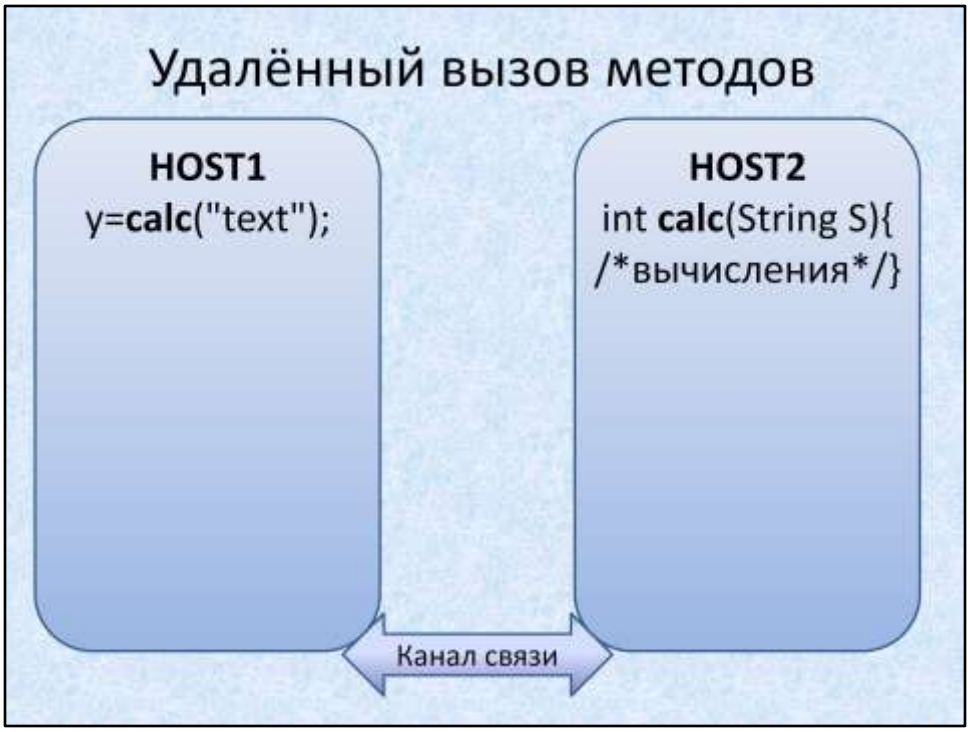


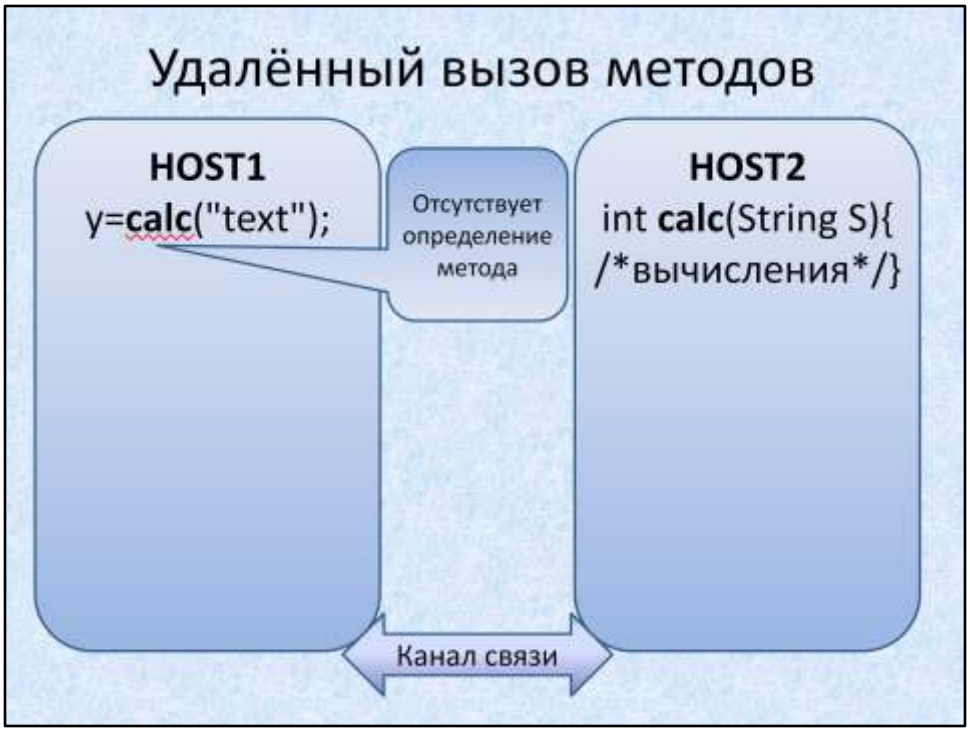
Меня зовут Полетаев Дмитрий Геннадьевич,
Тема дипломной работы «…»
Эта тема довольно обширна, над ней работало несколько человек, мы занимались созданием разных частей для решения одной большой задачи. О том, для чего создавалась эта система уже было рассказано, поэтому я не буду столь подробно на этом останавливаться, но если возникнут вопросы, вы можете задать их после доклада. Я же отмечу, что хотя эта система создавалась для упрощения проведения турнира программ, но задача объединения модулей, написанных на разных языках, может появляться довольно часто в больших проектах, т.е. в проектах в которых есть модули которые удобней написать на том или ином языке программирования.

Итак, у нас есть несколько программ, написанных на разных языках. В качестве топологии выбрана звезда, в центре которой находится интеграционный сервер. Каждая программа, участвующая в объединении, дополнена модулем, который обеспечивает взаимодействие с другими программами. Интеграционный сервер и протокол общения позволяют создавать между модулями виртуальные каналы

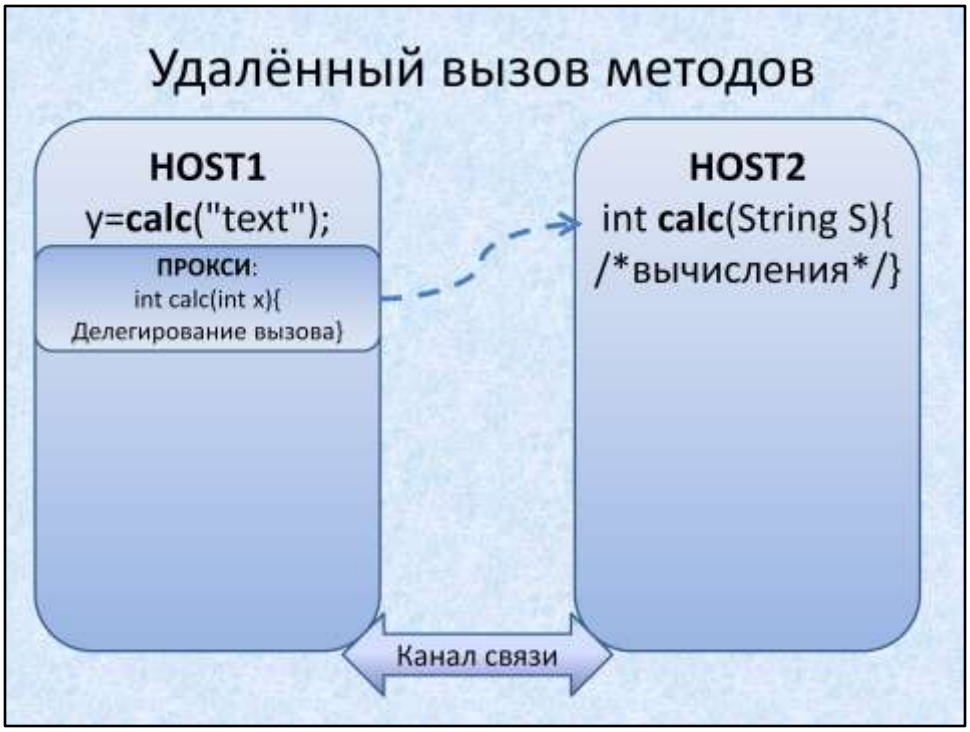
После этого можно сосредоточиться на объединении программ в предположении наличия между ними виртуального соединения «точка-точка».



Рассмотрим более детально задачу удалённого вызова метода. Пусть есть два компьютера связанные каналом. На компьютере Host2 расположен метод calc, и мы хотим обеспечить возможность вызова этого метода с компьютера Host1. Для осуществления удалённого метода необходимо решать задачи преобразования типов и синхронизации потоков вычисления на удалённых компьютерах. Продемонстрируем созданную архитектуру направленную на решение этих задач.

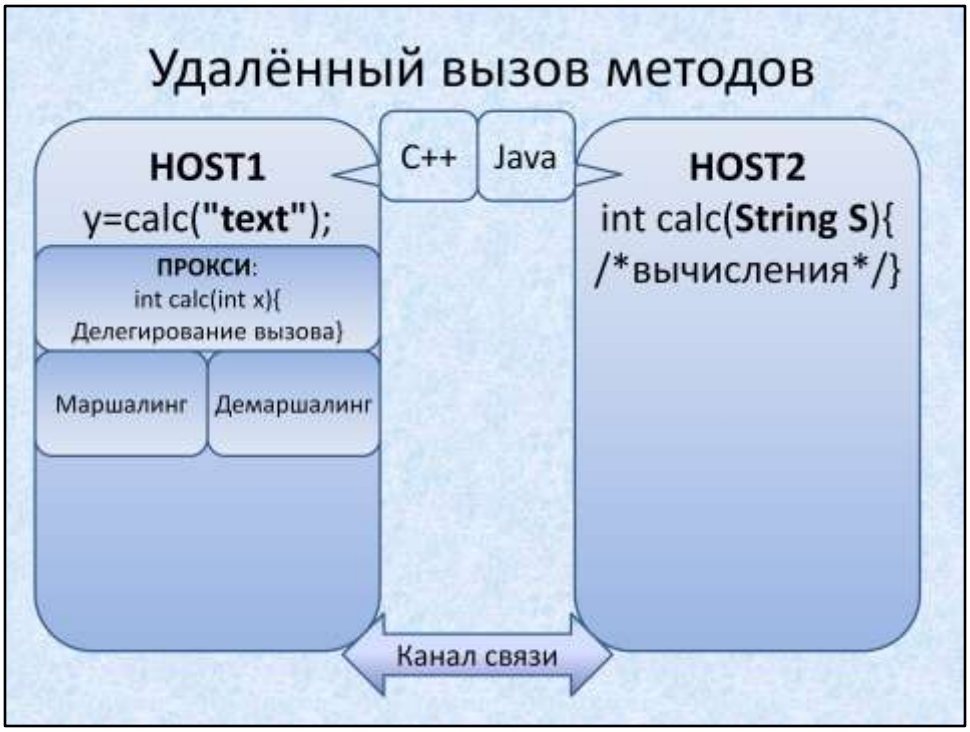


Первая проблема которая возникает – это отсутствие определения метода calc на компьютере Host1.



Эта проблема решается за счёт введения прокси метода, который являясь локальным методом для host1 позволяет вызывать удалённый метод используя синтаксис вызова локальных методов.

Следующая задача: компьютеры связаны каналом, который ориентирован на передачу потока байт или пакетов, а это означает что полученные прокси параметры нужно преобразовать в формат пригодный для передачи по сети.



Эта задача усложняется тем, что в объединении могут участвовать программы написанные на разных языках программирования, т.е. преобразовывать нужно в платформо-независимый формат.
Приведу пример: пусть вызывающий модуль написан на языке C++, тогда окавыченный текст является массивом типа char с нулевым завершением. Аналогичная конструкция на Java приведёт к созданию объекта String в кодировке UNICODE. Т.е. текс нужно во первых преобразовать из однобайтовой кодировки в уникод, а во вторых из массива байт в объект.
Маршалинг – это модуль, который выполняет преобразование параметров в независимый от языка формат, а демаршалинг выполняет противоположное преобразование.

Итак, после работы блоков преобразования типов, запрос необходимо передать на удалённую сторону, эту задачу выполняет диспетчер.

Итак, после работы блоков Прокси и маршалинга запрос превращается в массив байт, вся остальная работа по доставке запроса на удалённый компьютер, синхронизации потоков выполняется в Диспетчере.

Моя работа в данном проекте, кроме участия в проектировании, заключалась в разработке диспетчера для языка C++, т.е был написан код на языке C++, который выполняет эту работу.

Теперь, когда описаны основные модули участвующие в объединении, можно рассмотреть их взаимодействие.
При запросе удалённого метода вызывается прокси. С помощью маршалинга он упаковывает запрос и передаёт диспетчеру.
Диспетчер отвечает за доставку на удалённый хост, где сервис выполняет действия, обратные выполненным в прокси, и вызывает метод. После этого результат вычисления метода проходит в обратном направлении.

Даже из такой сильно упрощенной модели вызова можно определить список требований к диспетчеру.
Первое – возможность параллельно вызова диспетчера в многопоточном приложении, т.е. диспетчер должен уметь контролировать доступ множества потоков к общим ресурсам, например общему каналу передачи.

Второе требование – Мы хотим обеспечить запуск удалённого метода похожим на запуск локального.
При вызове локального метода в тот момент, когда управление возвращается в вызывавшую функцию результат метода уже вычислен.
Чтобы повторить тоже поведение для удалённых методов, диспетчер должен осуществлять блокировку потоков до тех пор, пока не будет получен результат вычислений от удалённой стороны.

Третье – диспетчер должен реализовывать сервер, готовый к получению пакетов, демультиплексировать единый канал для разных запросов.

Четвёртое – диспетчер должен поддерживать список предоставляемых сервисов и уметь перенаправлять запрос соответствующему сервису. Также диспетчер должен принимать решение о создании дополнительных потоков, т.к. современные многоядерные процессоры можно эффективно использовать только выполняя обработку в разных потоках.

И пятое – когда получен результат вычислений, необходимо извлечь из пула заблокированных потоков нужный поток.

В соответствии с перечисленными требования была спроектирован и реализован диспетчера. На этом слайде показано общее устройство модуля объединения, а также внутренняя архитектура диспетчера (справа).

В завершение доклада отмечу несколько моментов реализации. При проектировании диспетчера приходилось учитывать, что в последствии придётся портировать диспетчер на другие языки программирования и возможно платформы. Поэтому были созданы обёртки над системными вызовами. Кроме основой задачи улучшения переносимости они позволили во-первых упростить работу, а именно скрыть большую часть обработки ошибок и сделать интерфейс более удобным, а во вторых частично снять с разработчика заботу о освобождении занятых ресурсов.

При проектировании диспетчера было отмечено, что на канала передачи накладываются довольно слабые требования, а именно обеспечить двусторонний обмен данными. Это позволило выделить абстракцию канала, и реализовать сетевое соединения лишь как один частный случай канала передачи. Такое решение обеспечивает легкость замены транспорта, например с сокетов на pipe'ы

На этом мой доклад завершается, спасибо за внимание, вопросы?