Распределенные вычислительные системы

Лекция №3: ПО среднего слоя

Алексей В. Бурдаков, к.т.н.
burdakov@usa.net

План лекций

Дата	Тема
06.09	1. Вводная лекция: распределенные системы
13.09	2. Эволюция распределенных технологий, метамодель
20.09	3. Принципы ПО среднего слоя
27.10	4. Стандарты OMG, Архитектура CORBA, ORB
04.10	5. Пример приложения на CORBA
11.10	РК1
18.10	6. COM, Java/RMI
25.10	7. Определение местонахождения распределенных объектов
01.11	8. Долговременное хранение распределенных объектов
08.11	9. Распределенные объектные транзакции
15.11	10. Безопасность
22.11	11. Расширенное взаимодействие
29.11	РК2
06.12	Вакантно
13.12	Зачет/Экзамен

2-2

План лекции

- Особенности разработки распределенных систем
- Определение и типы ПО среднего слоя (middleware)
- Принципы RPC

2-3

Особенности распределенных систем с точки зрения проектировщика

- Ссылки
- Задержки заявки
- Активация/деактивация
- Миграция
- Постоянное хранение
- Одновременный доступ
- Связь
- Безопасность

2-4

Ссылки

- Ссылки на объекты в программных модулях на ОО языках программирования (например, C++) являются указателями в памяти
- Ссылки на объекты в распределенных системах в противоположность являются более комплексными:
  - Содержат информацию о размещении
  - Информацию о безопасности
  - Ссылки на объектные типы
- Ссылки на распределенные объекты значительно больше (40 байт для Orbix)

2-5

Задержки выполнения запросов

- Локальные вызовы требуют порядка пары сотен наносекунд
- Запрос к объекту требует от 0.1 до 10 миллисекунд
- Интерфейсы в распределенной системе должны быть спроектированы так чтобы снизить время выполнения запросов
  - Снизить частоту обращения
  - Укрупнить выполняемые функции

2-6

Активация/деактивация

- Объекты в ОО языках находятся в виртуальной памяти от создания до уничтожения
- В распределенных системах
  - Больше объектов
  - Объекты могут не использоваться на протяжении долгого времени
- Реализации распределенных объектов
  - Переносятся в память при активации
  - Удаляются из памяти при деактивации

2-7

Активация/деактивация

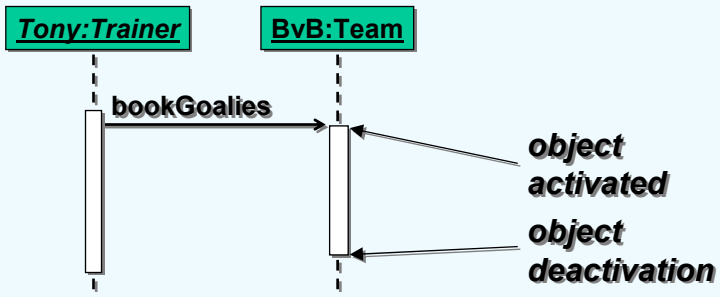


Диаграмма последовательностей (sequence diagram) UML

2-8

Активация/деактивация

- Вопросы, которые должны быть решены:
  - Репозитарий реализаций
  - Связывание объектов и процессов
  - Явная и неявная активация
  - Когда можно деактивировать объекты
  - Как обрабатывать одновременные запросы
- Кто должен решать эти вопросы?
  - Проектировщик
  - Разработчик
  - Администратор
- Как документировать решения?

2-9

Постоянное хранение

- Объекты имеющие и не имеющие состояния
- Объекты имеющие состояние должны сохранять его на постоянный носитель между:
  - Деактивацией объекта
  - Активацией объекта
- Может быть достигнуто
  - Экстернализацией на файловую систему
  - Отражением на реляционные БД
  - С помощью объектных БД
- Должно приниматься во внимание в процессе проектирования

2-10

Параллельное исполнение

- В нераспределенных системах исполнение в основном последовательное, иногда конкурентное в разных нитях процессов
- Распределенные компоненты выполняются параллельно, что приводит к необходимости согласования выполнения

2-11

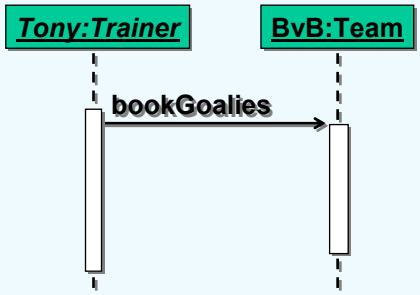
Взаимодействие

- В нераспределенных системах вызовы являются синхронными
- Альтернативы для распределенных объектов:
  - Синхронные
  - Односторонние
  - Отложенные синхронные
  - Асинхронные запросы
  - Множественные заявки

2-12

Взаимодействие

- Односторонние
- Асинхронные
- Отложенные синхронные
- Синхронные



2-13

Отказы

- Запросы в распределенных системах имеют большую вероятность отказов
- Клиенты обязаны проверять факт выполнения запросов сервером

2-14

Безопасность

- Безопасность в ОО приложениях может выполняться на основе контроля сеансов
- Распределенные объекты:
  - Кто запрашивает выполнение операции?
  - Как мы можем удостовериться, что субъект является именно тем за кого он себя выдает?
  - Как мы примем решение предоставлять или нет субъекту право на выполнение сервиса?
  - Как мы можем неопровержимо доказать, что сервис был предоставлен?

2-15

План лекции

- Особенности разработки распределенных систем
- Определение и типы ПО среднего слоя (middleware)
- Принципы RPC

2-16

Эталонная модель взаимодействия открытых систем ISO/OSI

2-17

Прямое использование сетевого протокола

- Ручное отображение комплексных параметров запросов в последовательности байтов
- Ручное разрешение проблемы гетерогенности представления данных
- Ручная идентификация компонентов с помощью указания доменных имен, номеров портов и т.п.
- Ручная реализация процедур активации компонентов
- Не гарантирована безопасность типов
- Ручная синхронизация взаимодействия между компонентами
- Качество сервиса не гарантировано

2-18

Концептуальный разрыв

- Объектная заявка: запрос на выполнение операции удаленного объекта
- Обмен данными реализован во всех современных сетевых ОС
- Концептуальный разрыв между вызовом операции и передачей данных
- Средний слой ликвидирует концептуальный разрыв

2-19

Эталонная модель ISO/OSI: ПО среднего слоя

2-20

ПО среднего слоя

- Размещается между приложениями и ОС/сетью
- Делает распределение прозрачным
- Решает проблемы гетерогенности:
  - Аппаратного обеспечения
  - Операционных систем
  - Сетей
  - Языков программирования
- Обеспечивает среду разработки и исполнения для распределенных систем

2-21

Виды ПО среднего слоя

- Транзакционно-оринетированное (2PC, DTP / XA)
  - IBM CICS
  - BEA Tuxedo
  - Encina
- Ориентированное на сообщения
  - IBM MQSeries
  - DEC MessageQueue
  - NCR TopEnd
- Системы, основанные на RPC
  - ANSA
  - Sun ONC
  - OSF/DCE

2-22

Виды ПО среднего слоя

- Объектно-ориентированное
  - OMG/CORBA
  - DCOM
  - Java/RMI

2-23

План лекции

- Особенности разработки распределенных систем
- Определение и типы ПО среднего слоя (middleware)
- Принципы RPC

2-24

Принципы RPC

- RPC – основа ПО среднего слоя
- Позволяет осуществлять вызовы процедур за границами узлов
- Интерфейсы вызовов определены с помощью IDL (Interface Definition Language)
- Компилятор RPC генерирует реализацию слоя представления и сессий из IDL

2-25

Пример RPC IDL

```
const NL=64;
struct Player {
 struct DoB {int day; int month; int year;}
 string name<NL>;
};
program PLAYERPROG {
 version PLAYERVERSION {
  void PRINT(Player)=0;
  int STORE(Player)=1;
  Player LOAD(int)=2;
 }= 0;
} = 105040;
```

2-26

Задачи, решаемые реализацией уровня представления

- Преобразование между прикладным и транспортным представлением данных называется маршалингом и демаршалингом

2-27

Маршалинг и демаршалинг

- Маршалинг: собирает данные в форму для передачи

```
char * marshal() {
 char * msg;
 msg=new char[4*(sizeof(int)+1) +
              strlen(name)+1];
 sprintf(msg,"%d %d %d %d %s",
         dob.day,dob.month,dob.year,
         strlen(name),name);
 return(msg);
};
```

2-28

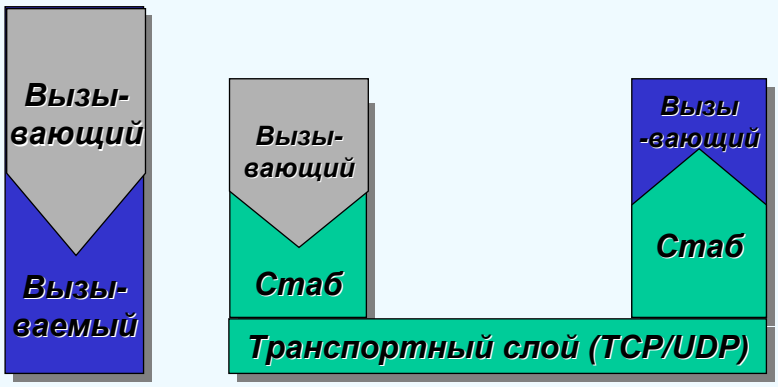
Маршалинг и демаршалинг

- Демаршалинг: разбирает данные в комплексные структуры

```
void unmarshal(char * msg) {
 int name_len;
 sscanf(msg,"%d %d %d %d ",
        &dob.day,&dob.month,
        &dob.year,&name_len);
 name = new char[name_len+1];
 sscanf(msg,"%d %d %d %d %s",
        &dob.day,&dob.month,
        &dob.year,&name_len,name);
};
```

2-29

Вызов метода и объектная заявка



2-30

Стабы (Stubs)

- Создание кода для маршалинга и демаршалинга трудоемко и ведет к ошибкам
- Код может быть сгенерирован полностью автоматически из определения интерфейсов
- Код внедрен в стабы клиентов и серверов
- Клиентский стаб представляет сервер для клиента, а серверный – клиентский для сервера
- Стабы безопасны с точки зрения типов
- Стабы также выполняют синхронизацию

2-31

Синхронизация (Synchronization)

- Цель: достигнуть такой синхронизации как и в случае локальных вызовов
- Достигается стабами:
  - Клиентский стаб посылает запрос и ожидает до тех пока сервер не вернет ответ
  - Сервер ожидает вызовов от серверного стаба и вызывает сервер когда приходит запрос

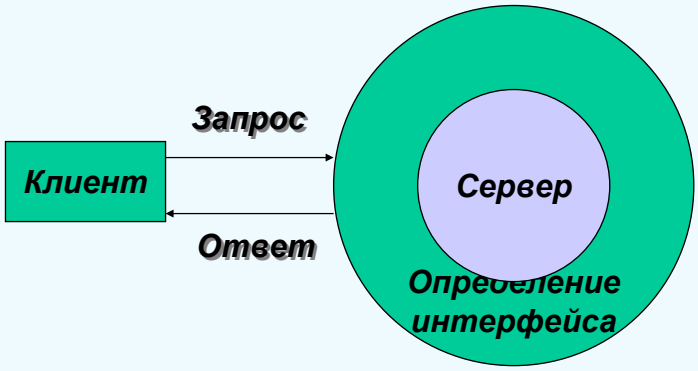
2-32

Безопасность типов

- Как мы можем гарантировать, что:
  - Серверы могут выполнить операции, запрошенные клиентами?
  - Параметры, переданные клиентом совпадают с ожидаемыми сервером?
  - Результат, передаваемый сервером совпадает с ожидаемым клиентом?
- ПО среднего слоя выступает в роли посредника между клиентом и сервером для обеспечения безопасности типов
- Достигается с помощью определения интерфейса на общем языке

2-33

Обеспечение безопасности типов



2-34

Литература / Internet источники

- В. Эммерих ***Конструирование распределенных объектов***. - М.:Мир. - 2002.
- М.Р. Когаловский ***Энциклопедия технологий баз данных***. - М.: ФС. - 2002.
- Ю.А. Григорьев, А.Д. Плутенко. - ***Жизненный цикл проектирования распределенных баз данных***. - Благовещенск. - 1999.
- Э. Таненбаум, М. Ван Стеен ***Распределенные системы. Принципы и парадигмы.*** – СПб.: Питер, 2003. – 877 с.: ил. – (Серия «Классика computer science»)
- www.omg.org

2-35