Распределенные вычислительные системы

Требования к распределенным системам и эволюция распределенных технологий

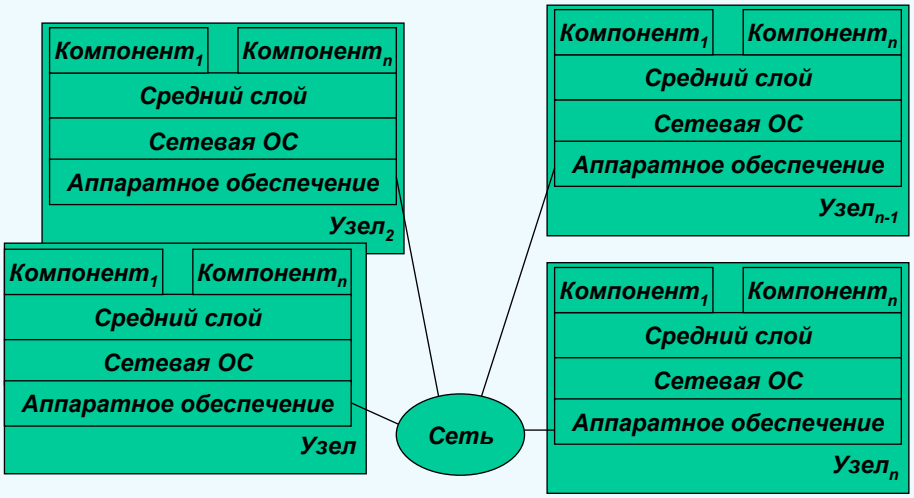
Алексей В. Бурдаков, к.т.н.
burdakov@usa.net

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

1-2

Что такое распределенная система?



1-3

Распределенная система: определение

Распределенная система - это совокупность автономных узлов, соединенных в вычислительную сеть. Каждый узел выполняет прикладные компоненты и ПО «среднего слоя», позволяющее компонентам координировать свои действия, так что пользователи воспринимают систему как единое целое.

1-4

Требования к распределенным системам

- Функциональные
- Нефункциональные (качественные)
  - Масштабируемость (scalability)
  - Открытость (openness)
  - Поддержка неоднородности (heterogeneity)
  - Разделение ресурсов (resource sharing)
  - Отказоустойчивость (fault tolerance)
  - Прозрачность (transparency)

1-5

Масштабируемость

- Адаптация распределенных систем к:
  - увеличению числа пользователей
  - уменьшению времени отклика
- Обычно осуществляется увеличением количества обрабатывающих единиц (процессоров, узлов, дисков, ….)
- Компоненты не должны изменяться при масштабировании
- Архитектура остается неизменной

1-6

Открытость

- Вопрос открытости вызван необходимостью изменения, расширения системы
- Необходимо описание синтаксиса и семантики компонентов системы
- Детализированные интерфейсы компонентов должны быть опубликованы (документированы)
- Должны быть решены различия в способах представления данных на различных платформах

1-7

Неоднородность

- Неоднородность
  - Операционных систем
  - Аппаратных платформ
  - Вычислительных сетей
  - Языков программирования

1-8

Разделение ресурсов

- Возможность свободного использования ресурсов в системе (аппаратного, программного обеспечения и данных)
- Менеджер ресурсов контролирует доступ, схему именования и управляет совместным доступом
- Возникает вопрос защиты и разделения доступа

1-9

Отказоустойчивость

- Сбои в ПО, аппаратном обеспечении, в СПД происходят!
- Распределенная система должна обеспечивать доступность сервисов даже при низкой надежности компонентов
- Отказоустойчивость достигается
  - Избыточностью (репликация, RAID-5)
  - Восстановлением (журнал БД)

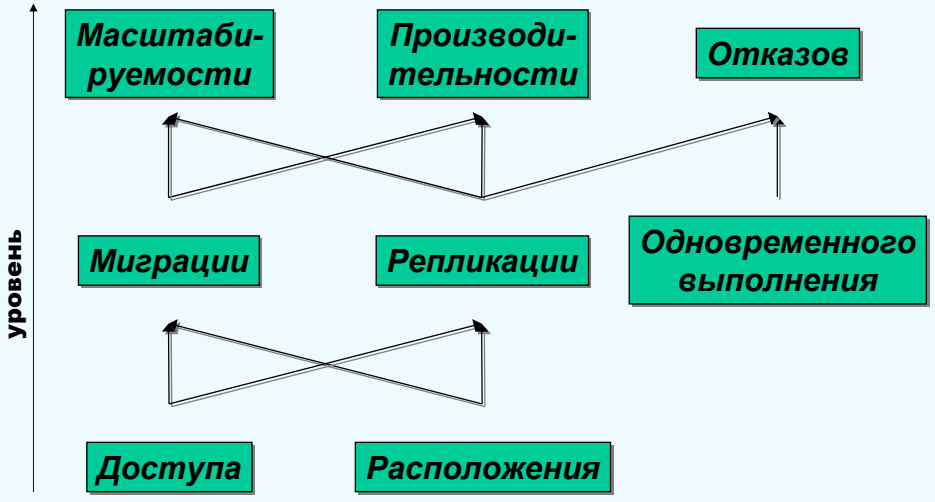
1-10

Прозрачность

- **Прозрачность (transparency)** – характеристика свойства данных или компьютерной среды, указывающей на то, что это свойство не воспринимается пользователем (программой) и может игнорироваться.
- Распределенная система должна восприниматься как пользователями, так и программистами как единое целое, а не как простой набор компонентов
- Вопрос прозрачности имеет несколько аспектов (составная часть Международного стандарта по открытой распределенной обработке ISO 20746-2 1996). Данные аспекты определяют те свойства, которые должны иметь распределенные системы

1-11

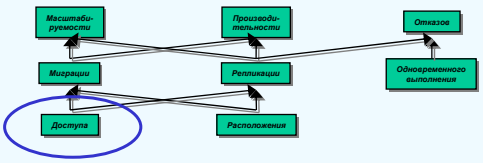
Прозрачность



1-12

Прозрачность доступа (access)

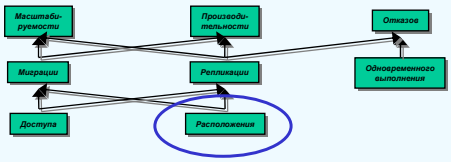
- Обеспечивает доступ к локальным и удаленным объектам единым образом
- Примеры:
  - Навигация в Web (Браузер, HTTP)
  - SQL запросы к распределенной БД (SQL)
  - Файловые операции в NFS



1-13

Прозрачность местонахождения (location)

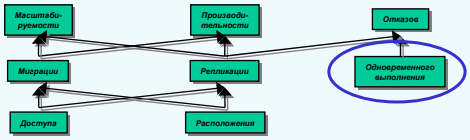
- Обеспечивает доступ к объектам без знания их местоположения
- Примеры:
  - Страницы Web
  - Таблицы в распределенной БД
  - Файлы NFS



1-14

Прозрачность одновременного выполнения (concurrency)

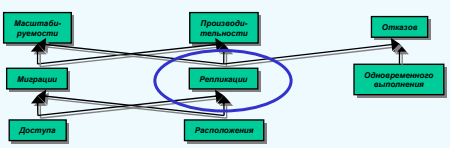
- Позволяет независимым процессам получать доступ к разделенным ресурсам одновременно без интерференции и с сохранением целостности
- Пример:
  - NFS
  - Сеть банкоматов
  - СУБД – забота СУБД обеспечить целостность как при единичном, так и одновременном доступе



1-15

Прозрачность репликации (replication)

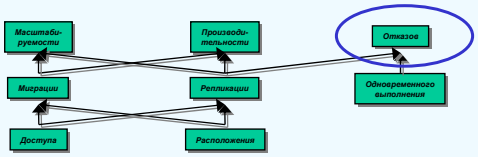
- Использование нескольких экземпляров информации для увеличения надежности и производительности. При этом ни для пользователей ни для приложений не известно с чем осуществляется взаимодействие (с копией или оригиналом).
- Пример:
  - Распределенные СУБД
  - Репликация Web-сайтов: round-robin, и др.



1-16

Прозрачность отказов (failure)

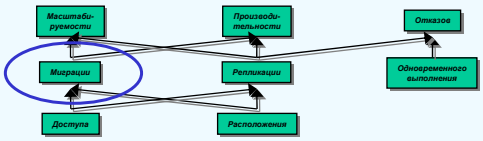
- Сокрытие отказов от пользователей, а также от клиентских и серверных компонентов
- Позволяет пользователям и приложениям завершить выполнение операций вне зависимости от сбоев и отказов других компонентов
- Пример:
  - Вызов сервиса в CORBA



1-17

Прозрачность миграции (migration)

- Возможность перемещения информационных объектов в системе без влияния на работу пользователей или приложений
- Примеры:
  - NFS
  - Страницы Web



1-18

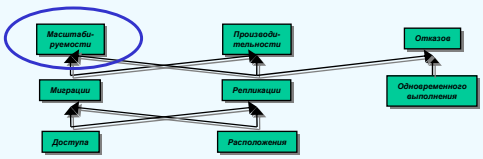
Прозрачность производительности (performance)

- Динамическая реконфигурация системы в зависимости от изменения нагрузки (балансировка нагрузки)
- Трудно достижима (трудно в априори оценить нагрузку)
- Примеры:
  - СУБД с балансировкой нагрузки

1-19

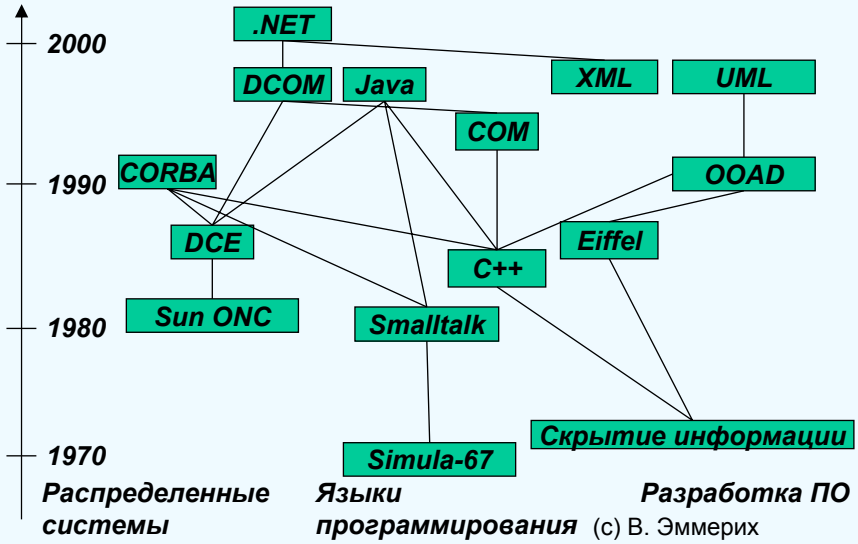
Прозрачность масштабирования (scalability)

- Позволяет масштабировать систему без изменения системной структуры и алгоритмов приложений
- Примеры:
  - WWW



1-20

Эволюция объектной технологии



(c) В. Эммерих

Эволюция объектной технологии

- **Simula** (1966, Скандинавия) - начало объектной ориентации
  - разработка программ имитационного моделирования,
  - понятие класса как типа,
  - экземпляры могут реагировать на сообщения
- **Скрытие информации** (1972, Д. Парнас)
  - сокрытие структур данных наиболее часто изменяемых при неизменном интерфейсе
  - идея модуля (module)

Эволюция объектной технологии

- **Smalltalk** (конец 70-х, начало 80-х) - для создания нового пользовательского интерфейса в центре Xerox
  - концепция - «все есть объект»,
  - наследование,
  - Полиморфизм
- **C++** (середина 80-х) – Бьярн Строструп вместе с группой исследователей из AT&T Bell Labs
  - строгое супермножество C - приемственность
  - статическое определение типов означающее контроль на стадии компиляции
  - более успешен нежели Smalltalk

1-23

Эволюция объектной технологии

- **RPC** разработанная Sun Microsystems как часть архитектуры **ONC** (Open Network Communications - Открытое сетевое взаимодействие):
  - удаленный вызов процедур,
  - описание интерфейсов с помощью специализированного языка,
  - использование клиентских и серверных стабов (stub),
  - передача параметров по сетевым протоколам
- **DCE** (Distributed Computing Environment - Среда распределенных вычислений) - стандарт OSF (Open Software Foundation);
  - стандарт RPC был включен в DCE
  - среди прочего определяет внешнее представление данных (External Data Representation, XDR),
  - служба высокого уровня: 1) имен и 2) безопасности

1-24

Эволюция объектной технологии

- **CORBA** – Common Request Broker Architecture – Общая архитектура брокера объектных запросов
  - версия 1.0 принята в 1991 г.
  - удаленный вызов и объектно-ориентированная парадигма
  - брокер объектных запросов – средний слой, обеспечивающий удаленные вызовы, независимость от расположения объектов, аппаратных и программных средств, языков прогр.
- **Java/RMI** (Remote Method Invocation - Удаленный вызов методов)
  - удаленный вызов методов интегрированы в Java
  - компоненты Java могут мигрировать и выполняться на различных платформах
  - нет проблем с неоднородностью, так как «Весь мир - это Java»

1-25

Эволюция объектной технологии

- **COM** (Component Object Model - Компонентная модель объектов) - модель взаимодействия приложений, написанных на разных языках
  - несколько интерфейсов (множественное наследование)
  - Только локальное взаимодействие
- **DCOM** (Distributed Component Object Model - Объектная модель распределенных компонентов)
  - представлена в Windows NT 4.0
  - доступ к удаленным объектам по OSF/RPC
- **.NET** – набор технологий и архитектурных моделей, предложенный Microsoft
  - XML – eXtensible Markup Language
  - SOAP – Simple Object Access Protocol
  - Web-сервисы

1-26

Эволюция объектной технологии: сравнение

Программный аспект

- DCE не является объектно-ориентированным, независимым от средств разработки, не поддерживает состояние объектов;
- DCOM зависим от платформы, имеет слабую независимость от языков и не поддерживает состояние объектов.

Коммуникационный аспект

- DCE, DCOM и .NET не поддерживают методы вызовов: асинхронный, отложенный синхронный, передача сообщений;
- DCE кроме того не поддерживает динамических вызовов, а DCOM – свободу перемещения объектов в вычислительной сети.

1-27

Эволюция объектной технологии

Инфраструктура:

- DCE определяет намного более слабую инфраструктуру, DCOM не определяет стандарта на инфраструктуру.
- .NET определяют только сервис именования UDDI

Признание и поддержка индустрией:

- CORBA – наибольшая поддержка.

1-28

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