



## 8. Разпределени софтуерни архитектури

Васил Георгиев

 v.georgiev@fmi.uni-sofia.bg  
 ci.fmi.uni-sofia.bg/

## Съдържание

- Модели на разпределени софтуерни архитектури
- Клиент-сервер, двуслойни, трислойни и  $n$ -слойни модели.
- Сервери за приложения и Web-сервери

## Обзор на РСА

- терминални комплекси [с минимашини] – mainframe architectures
- разпределени файлови системи
- клиент-сервер
  - двуслойни РСА
  - трислойни РСА
  - $n$ -слойни РСА
- метасистеми
- грид-системи
- сервисно-базирани архитектури (SOA)
  - Web-услуги

## Терминални комплекси

- централизирано обслужване на множество потребители с минимашина
  - терминали за потребителския интерфейс (текстови или графичен)
    - текстови терминали
    - интелигентни терминали – за графичен интерфейс
    - микротерминали: автономни персонални устройства
  - преход към клиент-сервер обслужване – 8.4
    - терминал
    - графична станция
- |                                |                       |
|--------------------------------|-----------------------|
| ➤ документен редактор          | Отворени системи с    |
| ➤ приложение и файлова система | ПК-базирани терминали |
| ➤ Web браузър                  |                       |

## Разпределени файлови системи

- работно сърверно пространство за разпределен достъп до файлове
- обработката се извършва само **локално** върху заредения контекст
- модел, приложим при
  - нисък темп на заявки към работното сърверно пространство (типично до няколко десетки потребители)
  - и относително автономна работа на клиентите с резервирана част от контекста

## Клиент-сървер архитектура за данни

- развитие на разпределените файлови системи за преодоляване на функционалните и нефункционалните им ограничения
- база данни замества файловия сървер
- обикновено имплементация на релационния модел бази
- редуциране на трафика чрез изпълнение на справки в базата вместо трансфер на цели файлове
- най-разпространени комуникационни протоколи за обмена между клиента и сървера са [първоначално] RPC и SQL (standard query language)

## Модел клиент-сървер

- разпределената обработка, при която проектират услуги (сървери), или процеси, които ползват тези услуги в режим заявка-отговор (клиенти).
- обикновено сърверите са:
  - проектирани да работят в режим на очакване на заявки за услугите, които предоставят
  - може да буферизират постъпващите заявки
  - не е задължително да връщат резултат – напр. заявките за негарантирана комуникационна услуга от предаващия клиент
  - N.B. Разпределеното приложение може да е организирано по модела клиент-сървер без да има ясно разграничаване на функциите на клиента и сървера:
    - сърверен процес може да е клиент на друг сървер от по-ниско ниво – йерархия на сърверите (респ. клиентите)
    - няколко сърверни процеса може да взаимодействат (комуникират) по модела на равнопоставените комуникации (т.е. всеки към всеки – peer-to-peer, p2p – напр. DNS)

## Комуникационни модели в клиент-сървер архитектурите

- алтернативи на RPC/RMI
  - Object Request Brokers (ORB)
  - Distributed Computing Environment (DCE)
  - Message-Oriented Middleware (MOM)
  - COM/DCOM
  - 3-слойни архитектури

## Слоести клиент-сървер архитектури

- клиентската и сърверната част обикновено се разделят за да се постигне паралелизъм и специализация на ресурсите.
- двуслойна архитектура – вж. фиг. 8.4 – проектирането на архитектурата се състои в избора на различни точки на разделяне на клиентската и сърверната част (т.е. между потребителския интерфейс и СУБД)
  - ефективна среда за групови приложения типично в *интранет* и *до няколко десетки потребители/клиенти*
- трислойна архитектура – със сървер на заявките – вертикално разпределение т.е. по йерархията на функциите (както вертикалното фрагментиране – по колони – при релационните БД)
- съвременни архитектурни подходи – хоризонтално разпределение – разпределение между логически еквивалентни части, обслужващи различна част от контекста на приложението – примери
  - репликирани Web услуги (mirror sites – консистентност!)
  - други Интернет-базирани репликирани услуги
  - р2р приложения

## Клиентски компоненти

- [G]UI
- команден интерпретатор (обикн. интегриран с GUI)
- прозрачно управление на достъпа до отдалеченото обслужване – напр.:
  - зареждане и изпълнение на сърверен стъб – достъп до интерфейсите на отдалечен обект като на локален обект
  - поддържане на прозрачна адресация – напр. чрез конвенционална система от символни имена
  - управление на достъпа до група обекти (особено реплики) чрез разделяне на единния поток потребителски заявки и сливане на отговорите – 8.10
  - нефункционални черти – напр. избор на най-«близък» сървер

## Сърверна архитектура

- итеративни сървери
  - сърверният процес е един за всички
  - обслужва последователно клиентските заявки и евентуално връща резултат
  - при необходимост ги буферира
  - заявките изчакват обслужване
- конкурентни сървери
  - сърверният процес приема заявките и ги предава за изпълнение на друга нишка или процес – т.е.:
    - многонишков сървер, който стартира нова нишка за всяка заявка
    - създаване на процес-реплика с **fork** в UNIX
- сърверни портове
  - клиентските заявки се предават до точките на обслужване – ports/endpoints, които се сканират от сърверния п-с за заявки
  - един сървер се асоциира с един порт (+ IP адрес или форматно име)
  - статични портове – стандартизация: IANA (Internet Assigned Numbers Authority) FTP сървер на TCP-p.21, HTTP сървер на TCP-p.80
  - динамични портове (за нестандартизираните сървери)
    - резидентен списък: ОС поддържа демон за динамичен списък на портове (DCE - Distributed Computing Environment) – 8.11
    - суперсървери - конкурентен сърверен демон - inetd в UNIX за Интернет-услугите - приема всички заявки и стартира с fork процес за обслужване
      - редуцира се системния свръхтовар на обслужване, понеже няма неактивни чакащи заявка процеси

## Клиентски контекст в сърверите

- сървери с поддържане на клиентския контекст (stateful) – преобладаващо приложение -
  - сърверът поддържа таблица с двойките клиент-контекст (напр. клиент-файл-режим)
- сървери без поддържане на клиентския контекст (stateless)
  - [HTTP] cookies: пакет информация, зареден от уеб сървър към Интернет браузър, а след това връщан при всяка следваща заявка
    - емулиране на сесия: част от процеса на достъп, сесияен контекст (напр. «пазарска кошница»)
    - персонализиране на настройките (изглед на сайт)
    - клиентски дневник (log) – напр. история на посещенията
  - модифициране на сърверните имена (URL rewriting или rewrite engine)
    - напр. `http://example.bg/index.php?title=Page_title` → `http://example.bg/Page_title`
    - освен за съкратен URL се ползва и за сесийни персонализирани страници за достъп на клиентите – когато не се поддържат cookies:
      - напр. `http://example.bg/index.jsp` → `http://example.bg/index.jsp;jsessionid=ABC`
  - скрити полета в HTML-форми [hidden fields] – с wizard (за колекция от няколко форми за промяна на зависим полета в БД)
    - напр. `<input type="hidden" name="EXPIRED" value="1200">`

## Ограничения на двуслойните СА

- по производителност:
  - ефективна среда за групови приложения типично в **интранет** и **до няколко десетки потребители/клиенти**
  - ограничена скалируемост поради необходими комуникации за поддържане на отворена сесия между централизирания сървер и всички клиенти (дори и когато няма потребителски обмен)
- по функционалност:
  - приложимост предимно за СУБД-базирани приложения
  - ограничена преносимост при смяна на сърверите

## Трислойен модел на СА

- въвежда на междинен слой (middleware) между клиентския потребителски интерфейс и сърверния контролер на базата данни – 8.14
- подходи за реализация на междинния слой:
  - монитори за транзактивно обслужване (Transaction Processing – TP)
  - MOM
  - сървери за приложения
  - Web-сървери
- предимства на приложението на междинен слой
  - асинхронност на клиентския и сърверния процес (ако се поддържа буфериране на съобщенията-заявки и на резултата)
  - възможност за планиране на сърверното обслужване на клиентските заявки по време, приоритет, ресурсна обезпеченост и др.
  - ефективно приложение за групово обслужване на хиляди клиентски процеси
  - по-голяма преносимост и модулност (адаптивност): клиентските и сърверните приложения се проектират независимо

## Йерархия на модела клиент-сървер

- три нива на процесите, към които се отнасят съответно сърверните и клиентските процеси:
  - ниво на потребителския интерфейс – обслужва предимно графични и други интерфейси
  - ниво на обработката – състои се предимно от програмната имплементация на обслужващите алгоритми
  - ниво на данните – обслужва достъпа до ползваните данни
- нивото на потребителския интерфейс обикновено съдържа процеси-клиенти, които поддържат най-често сложен графичен интерфейс, но преобразуват графичните команди в приложни данни или команди, които се интерпретират и предават към съответните сърверни процеси – напр. влачене на файл към коша за буклук
- нивото на процесите може да е с примитивна или сложна архитектура – примери
  - търсачка в Интернет – 8.15
  - НЕР - симулатор
  - офис-приложение, базирано на локален файл-сървер и комуникационни примитиви
- нивото на данните – процеси-сървери – напр:
  - файлови сървери или релационни БД, които са клиенти за файловите сървери
  - обектни данни (данни + методи) в обектни бази данни – напр. при симулатори
  - CAD системи
  - мултимедийни приложения

## Трислойна технология

- трите слоя (освен като клиент-мидълуер-сървер) се разграничават също и като – 8.16
  - представителен слой
    - за потребителски-ориентирани услуги
    - напр. браузър, изпълнява се от настолни платформи или микротерминали
  - логически слой
    - функционално разделяне на приложението – напр.
      - достъп до съдържание
      - числово-логическа обработка – напр. електронни таблици, симулатори, CAD-приложения
      - формиране на интегрален резултат от няколко сърверни приложения
      - разделянето на логическия междинен слой по функции, технологии или друг признак се означава като **многослойна** или **n-tier** СА
    - напр. сървер за приложения - JEE; поддържа се от сърверни платформи на работни станции - вкл. разпределени сърверни приложения
  - слой данни - поддържа се от релационни СУБД на минимашини (mainframes)
- вж. пример за интегрирана 3t-платформа SAP R/3  
([http://en.wikipedia.org/wiki/SAP\\_R/3](http://en.wikipedia.org/wiki/SAP_R/3))

## 3tCS с TP монитори

- транзактивния монитор е буфер на заявките (т.е. специализиран MOM), който извършва:
  - [буфериране на съобщенията-заявки]
  - планиране:
    - на конкурентните транзакции (вж. л-я 11.)
    - по приоритети
- транзакцията се обслужва от монитора – асинхронно на клиентския процес - 8.17
- разширена функционалност и нефункционални параметри (c/o 2tCS) – т.е. области на приложение:
  - промени в няколко СУБД с една транзакция
  - прозрачност (от страна на клиента) към сърверната технология (разпределена файлова система, СУБД, не-релационни бази)
  - приоритетност и конкурентност на транзакциите
  - възможност за защита в отворена система/интернет
  - висока скалируемост и ефективно натоварване на сърверната инфраструктура - вкл. ефективно използване на вторични и третични памети
- имплементации:
  - вградени ТМ в СУБД – ограничена производителност – считат се за разновидност на 2tCS
  - ТМ извън СУБД – типичен 3tCS, но необходимост от допълнително технологично поддържане в сърверната част

## Функции на TP мониторите

- клиентски функции на TP мониторите:
  - поддържат достъпа и обслужването на [хиляди] клиентски процеси:
    - асинхронно
    - без предистория (stateless) – т.е. безсесийно – транзакция по транзакция
    - прозрачно за СУБД (СУБД "вижда" само единен сериен поток от заявки)
  - приложими за реализиране на междуклиентски комуникационни модели - напр. препредаване (Store-and-forward); асинхронен обмен; сесия
- системни функции:
  - планиране и балансиране на сърверното натоварване
  - консистентност на данните
  - отказоустойчивост (рестартиране на пропаднали процеси)
  - скалируемост – чрез добавяне на нови сървери

## Технологии за TP монитори

- Java Transaction API (JTA) е пакет в Java EE за поддържане на разпределено транзактивно обслужване
  - пакет javax.transaction - вж. поддържаните методи на <http://java.sun.com/jaavaee/5/docs/api/javax/transaction/package-summary.html>
- IBM ALCS (Air Line Control System) – <http://www-306.ibm.com/software/http/tpf/alcs/> – високо-производителен и надежден сървер за интензивно транзактивно обслужване в IBM-сърверна среда с OS/390 и z/OS и MQSeries

## 3tCS с MOM

- поддържа:
  - асинхронност между клиента и сървера
  - приоритетно мултиплексиране на множество клиентски порцеси
- предава заявките към СУБД под формата на съхранени съобщения
- съобщенията в MOM се идентифицират освен чрез адреси още с приоритет и идентификатор на всяко съобщение
- MOM (за разлика от TP мониторите, които третират транзакциите като фрагменти данни) поддържа обработка на съобщенията в транзит - напр.
  - преформатиране съдържание за различни платформи,
  - преформатиране съгласно клиентски заредими процедури -
    - прилага се при преобразуване на графични клиентски команди като влачване на обект (drag-and-drop) в последователност изпълними инструкции
- най-широко приложение за обслужване в мрежи със загуби – напр. безжични
- проблеми:
  - необходимост от специализирана сърверна инфраструктура - боркери и агенти на съобщенията
  - ограничена приложимост при синхронни (напр. ММ) и интерактивни CS приложения

## 3tCS със сървер за приложения

- логиката на обслужването е пренесена в сърверната част, клиентският процес поддържа само потребителския интерфейс (обикн. GUI) в т.ч. на микротерминали
- сервисната платформа се поддържа от разпределен хост за група клиенти и имплементира
  - приложната логика
  - числовата обработка
  - достъпа до данните
- пренася се поддръжката и имплементацията в сърверната част
  - преносими, многоплатформени и “тънки/леки” клиенти
  - улеснено управление – функциите по защита, настройка, осъвременяване се централизират
  - висока скалируемост, ниска обща себестойност на обслужването (TOC – total cost of ownership)

8. Разпределени СА

ФМИ/СУ

\* ИС/СИ

\* РИТАрх/РСА

21

## Сървери за приложения

- СП обслужват интерфейсите клиенти обикновено върху HTTP – като Web-сървери
- различават се от W-с по преобладаващата генерация на динамично съдържание и интензивни операции върху контекста в БД
- разпределеното приложение се реализира на базата на централизирани колекция от зависими услуги или модули в сърверната част – web-сървери, БД, CAD/CAM приложения
- СП често са достъпни чрез платформено-независим API
- клиентите се проектират само като интерфейс към вградените интерфейсни инструкции, интерпретирани от СА
- порталите са метод за изграждане на колективни/бизнес сървери за приложения с един Web-адрес за множество приложения
  - персонализиран но единен интерактивен (PB) достъп до разпределени сърверни ресурси и разпределени сърверни приложения
  - бизнес-порталите допълнително предоставят споделено работно пространство
  - портлет – сърверно приложение, вградено в клиентския интерфейсен прозорец (браузър), чието съдържание се представя в портала – тематичен/функционален компонент на различни портали – напр. бюлетин, чат, метеорологична справка,
- хоризонтални (yahoo) и вертикални (flysky.com) портали

8. Разпределени СА

ФМИ/СУ

\* ИС/СИ

\* РИТАрх/РСА

22

## Web сървери

- специализиран вариант на сървери за приложения, които обслужват HTTP-заявки от специализирани клиенти – Web браузъри и връщат HTTP-отговор
  - обслужва стандартно порт 80 на сърверната платформа
  - за HTTPS (с SSL/TLS) заявки порта е 443
- съответно и извлеченото или генерирано съдържание е предназначено за интерпретация от браузър
- съдържанието е основно HTML-текст, но може да съдържа изображения или друг документен или общ тип файл, дефиниран съгл. Multipurpose Internet Mail Extensions (MIME)
- записват последователните несвързани заявки на един клиент в log (дневник)
- освен статично съдържание (HTML страници), предоставят интерфейс към сърверни Web-приложения, ползвайки стандартен интерфейс/език/протокол – напр: CGI, SSI, JSP, PHP, ASP, ASP.NET
- редица специфични системни функции (контрол на трафика, копресия/аекомпресия на съдържанието, виртуално хостване – споделяне на един IP-адрес между няколко сайта)

8. Разпределени СА

ФМИ/СУ

\* ИС/СИ

\* РИТАрх/РСА

23

## Web приложения

- сърверни приложения, предназначени за интерпретация от Web-браузери - затова се проектират на съотв. езици - HTML, ASP, PHP, Perl
- използват се типично за достъп до лично (Webmail, Weblog - blog) или споделено работно пространство (wiki pages) или за публикуване на комунални услуги - електронна търговия и др.т.
- възможна частична интерпретация от клиента, базирана на Java, JavaScript, DHTML, Flash
- предимства:
  - базират се на общо интерфейсно приложение – браузър
  - автоматично/централизирано осъвременяване чрез сърверната част
  - платформена независимост
  - допускат ограничени права на потребителя (nomadic user)
- проблеми:
  - зависимост от мрежова свързаност, моментно натоварване и сърверна стабилност
  - производителност – неподходящи за критични или твърдо-PB приложения

8. Разпределени СА

ФМИ/СУ

\* ИС/СИ

\* РИТАрх/РСА

24

## Богати Web приложения

- Rich Internet applications (RIA) са
  - Web клиенти, поддържащи *локално* пълната функционалност на класически настолни приложения – CS/fat-client
  - основната част от контекста им се съхранява, обновява и записва на отдалечен сървер за приложения (като при CS/thin-client)
  - клиента обикновено е базиран на уеб-браузър – без инсталация на конкретно приложение
    - с начално (а и run-time) зареждане на интерпретиран код – аплети, client engine – който поддържа специализирания GUI и обмена със сървера
  - изпълняват се в специализирана защитена локална среда – т.нар. пясъчник (sandbox)
- предимства
  - пълна платформена преносимост и независимост (nomadic users)
  - автоматично централизирано обновяване (upgrade)
  - защитеност от конвенционални вируси
  - оптимизират комуникациите и производителността на приложението (c/o CS/thin-client)
    - изпреварващо предварително зареждане на контекст (prefetching)
    - асинхронни прозрачни комуникации
- проблеми
  - ограничения на достъпа извън пясъчника
  - интерпретиран код – скорост на обработка и зареждане

8. Разпределени СА

ФМИ/СУ

\*

ИС/СИ

\*

РИТАрх/РСА

25

## RIA – реквизити

- пясъчник – sandbox
  - защитен механизъм за интерпретиране на заредим код (аплети)
  - ограничен достъп до локалната памет – динамичен виртуелен диск (логически отделен от файловата система) за междинни данни – в първичната и вторичната памет
  - без достъп до мрежа и локални интерфейсни устройства (обмен през браузъра)
- имплементации на аплети (виртуализация, емуляция на платформи)
  - Java applets – прекомпилиран и компресиран байтов код на Java с общо предназначение, който се интерпретира от браузър върху JVM
  - JavaScript – слабо-типизиран не-обектен език с вградена интерпретация от браузъра (общо предназначение)
  - Adobe [Shockwave / Macromedia] Flash и MS Silverlight – език и интерпретатор на векторна (и растерна – за Flash) MM (анимация + звук) с поддръжка на поточни данни
  - ActiveX controls – [стандарт за проектиране на и библиотека от] компоненти за GUI, поддържани от IE и MSWindows, които включват командни бутони, интерфейсни полета, изборни списъци и др. – вкл. IE за вграждане

8. Разпределени СА

ФМИ/СУ

\*

ИС/СИ

\*

РИТАрх/РСА

26

## Упражнения по разпределени SW архитектури (м. май)

- 1) Софтуерни архитектури за поточни данни
- 2) Многоишкови разпределени приложения
- 3) Шаблони за разпределени SW архитектури  
EJB + Session beans, Web Service и MVC

8. Разпределени СА

ФМИ/СУ

\*

ИС/СИ

\*

РИТАрх/РСА

27