**8. Разпределени софтуерни aрхитектури**

**Обзор на РСА**

- терминални комплекси [с минимашини] – mainframe architectures

- разпределени файлови системи; клиент-сървер (двуслойни РСА; - трислойни РСА

*n*-слойни РСА) - метасистеми

- грид-системи сервизно-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 кл.-с. обслужване – 8.4фиг;

**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 RPC и SQL

**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 буферират 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: 1)сърверен процес може да е клиент на друг сървер от по-ниско ниво –йерархия на сърверите (респ. клиентите) или 2)няколко сърверни процеса може да взаимодеистват (комуникират) по модела на равнопоставените комуникации (т.е. всеки към всеки – peer-to-peer, p2p – напр. DNS)

**Слоести клиент-сървер архитектури**

Клиентската и сърверната част обикновено се разделят за да се постигне паралелизам 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)

- 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 – разпределе-ние между логически еквивалентни части, обслужващи различна част от контекста на приложението – примери:

репликирани Web услуги (mirror sites – консистентност!), 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 потреби-телски заявки и сливане на отговорите – 8.

**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:21, HTTP:80. Динамични портове (за нестандартизираните сървери) резидентен списък: ОС поддържа демон за динамичен списък на портове (DCE – Distributed Computing Environment)

**Клиентски контекст в сърверите**

сървери с поддържане на клинетския конткст (stateful) - преобладаващо приложение: сърв. поддържа таблица с двой-ките клиент-контекст (напр. клинет-файл-режим)

сървери без поддържане на клинетския 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, сесиен 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

- скрити полета в HTML-форми – с wizard (за колекция от няколко форми за промяна на зависими 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 неоходими 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

**Трислоен модел на СА -** въвеждане на междинен слой (middleware) между клиентския UI и сърверния контролер на базата данни – 8.14фиг.

Подходи за реализация на междинния слой: - монитори за транзактивно обслужване (Transaction Processing – TP)

- МОМ; - сървери за приложения

- Web-сървери

Предимства на приложението на междинен слой: 1. асинхронност на клиентския и сърверния процес (ако се поддържа буфериране на съобщенията-заявки и на резултата); 2. възможност за планиране на сърверното обслужване на клиентските

заявки по време, приоритет, ресурсна обезпеченост и др. 3.ефективно приложение за групово обслужване на хиляди клиентски процеси; 4. по-голяма преносимост и модулност (адаптивност): кл. 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 UI – обслужва предимно графични и други интерфейси; 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.

- ниво на обработката – състои се предимно от програмната имплементация на обслужващите алгоритми. Може да е с примитивна или сложна архитектура – пример. търсачка в Интернет – 8.15фиг.

- ниво на данните – обслужва достъпа до ползваните данни. Напр. файлови сървери или релационни БД, които са клиенти за файловите сървъри, CAD системи, мултимедийни приложения

**Сървери за приложения**

СП обслужват интерфейсните клиенти обикновено върху НТТР – 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 (РВ) достъп до разпределени сърверни

ресурси и разпределени сърверни приложения

**Web сървери**

специализиран вариант на сървери за приложения, които обслужват НТТР- заявки от специализирани клиенти – Web браузъри и връщат НТТР-отговор. Обслужва стандартно порт 80 на сърверната платформа. За HTTPS (с SSL/TLS) заявки порта е 443. 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:JSP, PHP, ASP, ASP .NET Редица специфични системни функции (контрол на трафика, компресия на съдържанието, виртуално хостване – споделяне на един IP-адрес между няколоко сайта)

**Web приложения**

сърверни приложения, предназначени за интерпретация от Web-браузери - затова се проектират на съотв. езици - HTML, ASP, PHP, Perl. Използват се типичино за достъп до лично (Webmail, Weblog - blog) или

споделено работно пространство (wiki pages) или за публикуване на комунални услуги - електронна търговия и др. 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

проблеми: зависимост от мрежова свързаност, моментно натоварване и сърверна стабилност; производителност – неподходящи за критични или твърдо-РВ приложения

**Богати Web приложения**

Rich Internet applications (RIA) са

Web клиенти, поддържащи *локално* пълната функционалност на класически настолни приложения. Осн.част от контекста им се съхранява, обновява и записва на отдалечен сървер за приложения. Клиента обикновено е базиран на уеб-браузър – без инсталация на конкретно. Изпълняват се в специализирана защитена локална среда – т.нар. пясъчник (sandbox).

Предимства: пълна платформена преносимост и независимост,

автоматично централизирано обновяване (upgrade), защитеност от конвенционални вируси, оптимизират комуникациите и производителността на приложението

проблеми - ограничения на достъпа извън пясъчника, интерпретиран код – скорост на обработка и зареждане

**RIA – реквизити**

пясъчник – sandbox

- защитен механизъм за интерпретиране на заредим код (аплети)

- ограничен достъп до локалната памет – динамичен виртулен 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, емулация 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) ММ (анимация + звук) с поддръжка на поточни данни

- ActiveX controls – [стандарт за проектиране на и библотека от] компоненти за GUI, поддържани от IE и MSWindows, които включват командни бутони,

интерфейсни полета, изборни списъци и др. – вкл. IE за вграждане