19. Интеграция на разпределени информационни системи

Нужда и дефиниция за разпределена система

За нарастващата нужда от разпределени системи се обуславя от няколко различни фактора:

- Нарастващи сливания между фирми
- Намалява се наличното време за осигуряване на необходима услуга
- Интернет осигурява нови възможности за предлагане на софтуер и услуги на много клиенти

Дефиниции

Таненбаум

Разпределената система представлява съвкупност от компютри, които изглеждат на потребителите си като една кохерентна система.

Емерих

Разпределена система е съвкупност от автономни хостове, които са свързани чрез компютърна мрежа. Всеки хост изпълнява компоненти и върху него работи разпределен middleware, който позволява компонентите да координират своите активности по такъв начин, че потребителите да възприемат системата като единично, интегрирано средство за работа.

Middleware

С усложняването на софтуера се налага и по- голяма абстракция и все по-сложни примитиви за разработка.Така се преминава от операционни системи за разпределени компютри, разпределени операционни системи към мрежови операционни системи докато се стига от нуждата от нов абстрактен слой, които да предоставя основни услуги и който има за цел прозрачност на разпределеността на системата.Този слой е middleware.

Дефиниции

Дефиниция 1:

Лепило, което държи всички компоненти на разпределените приложения заедно.

Дефиниция 2:

Компонент, който държи решението на проблема с разширяемостта чрез създаване на ниво в разпределената инфраструктура, върху което се разполагат приложенията.

Дефиниция 3:

Middleware е термин, който се отнася до множество от софтуерни услуги, което се намира между приложението и операционната система и има за цел да улесни разработването на разпределени приложения чрез абстрахиране от сложността и хетерогенността на намиращата се отдолу среда операционни системи, хардуерни платформи и комуникационни протоколи).

19. Интеграция на разпределени информационни системи

Функции

Middleware трябва осигурява примитиви на високо ниво, които опростяват конструирането на разпределена система Трябва да предоставя Консистентен опростен приложен програмен интерфейс (API).Трябва да спомага за оперативната съвместимост на хардуер, мрежи и програмни езици.Позволява на разработчиците да се фокусират върху изискванията , а не върху технологичната среда. Middleware-ът трябва да поддържа мрежова комуникация, да позволява синхронизация по различни начини, да се справя с хетерогенността, да бъде надежден и разширим.

Роля

За да се разбере ролята на middleware-а, трябва да бъде осъзната двойнствената му същност на **програмна абстракция** и на **инфраструктура.**

Програмна абстракция

- Има за цел да скрие детайлите от ниско ниво за мрежи, хардуер и разпределеност
- Има тенденция към все по мощни примитиви, които без промяната на концепцията на RPC, да предоставят повече свойства и по-голяма гъвкавост.
- Начинът за предоставяне на примитивите на разработчика зависят от различните програмни езици;

Инфраструктура

- Има за цел да достави изчерпателна платформа за разработка и изпълнение на сложни разпределени системи
- Има течения към SOA
- Използва се продуктовата линия на един доставчик на middleware за да се улесни разработката
- Тенденцията е към интеграцията на различните платформи и към гъвкавост чрез конфигурации

Видове Middleware

Remote procedure calls (RPC)

Осигурява инфраструктура за отдалечени извиквания на процедури. Служи за основа на почти всички останали форми на middleware.

Transaction processing monitors (TP monitors)

Могат да се разглеждат като RPC с възможност за транзакции. Класифицират се в TP-Lite (RPC интерфейс за бази от данни) и TP-Heavy (средства и функционалност подобни и дори надминаващи тези на операционните системи).

Обектни брокери

Повечето от тях използват RPC при имплементацията на отдалечено повикване на обекти.

Обектни монитори

Обединяване на TP мониторите и обектните брокери в хибридни системи.

19. Интеграция на разпределени информационни системи

Message-oriented middleware (MOM)

Осигурява транзакционен достъп до опашки,примитиви за четене и запис в локални и отдалечени опашки.

Брокери на съобщения

Осигуряват възможност за трансформиране и филтриране на съобщения при прехвърлянето между опашките. Динамично определят получателя на съобщението въз основа на съдържанието му.

Проблеми на интеграцията на софтуерни приложения с използването на обмен на файлове, по сокети, RPC и RMI;

Интеграция чрез обмен на файлове

Общо описание

Всяко приложение трябва да създава файлове , които другите приложения трябва да консумират.Интеграторите се грижат за това как файловете се трансформитат към различни формати.Файловете трябва да се създават на равни интервали според нуждите на бизнеса.

Положителни страни

- Не изисква промени в системите , които биват интегрирани.Подходът не е инвазивен.
- Ниско ниво на свързаност между интегрираните системи
- Сравнително прост метод

Отрицателни страни

- Неефективни при чест обмен на файлове
- Неконсистентност на данни, трудно се синхронизират файловете
- Трябва да се имплементират механизми за логическо заключване, да се използват сложни именуващи конвенции, проверки за актуалност на файла
- Проблем за местоположението на файла , когато се интегрират два бизнеса.

Интеграция чрез сокети

Общо описание

Сокет е една от крайните точки на двупосочна комуникация между две програми, които работят в мрежова обстановка.Всеки сокет е свързан с номер на порт , така че TCP слоят да може да идентифицира приложението, където информацията трябва да се предате

При интеграция софтуерни системи чрез сокети , трябва да се отбележи, чесе използват възможностите на TCP(Transmission control protocol) и UDP (User DataGram Protocol) протоколите.При този вид едната или повече от системите играят ролята на сървър и една или повече от системите играят ролята на клиент.Сървърът притежава сокет , които е свързан с определен порт на който слуша за съобщения. Клиентът трябва да знае името на хоста на машината , на която се намира сървъра и порта , на който сървърът слуша. За да осъществи връзка клиентът опитва да се свърже със

сървъра през хоста и порта на сървъра.Клиентът трябва да се индентифицира като му подава локален за клиента порт , които ще бъде използван по време на комуникацията.

Положителни страни

- Удобни за имплементация на клиент-сървър комуникация
- Едновременна поддръжка на много клиенти
- Ефикасна UDP комуникация

Отрицателни страни

- Високо ниво на свързаност
- Повечето системи използват собствени формати на данните
- Трябва да се обръща твърде много внимание на жизнения цикъл на комуникационните обекти

Интеграция с RPC

Общо описание

Remote procedure calls е технология за комуникация, която позволява на една програма да извика функционалност на друга, която действа в друго адресно пространство като най-често тя физически е разположена на друг компютър. RPC скрива детайлите при комуникацията зад процедурни извиквания и подпомага взаимодействието между хетерогенни платформи.Архитектура на RPC включва:

- Клиентски стъб: парче от код, което се компилира и свързва при клиента.
- Сървърен стъб: имплементира сървърната страна на повикването.
- Кодови шаблони и референции: IDL компилатора създава всички необходими помощни файлове; може да генерира и шаблони с първичен код за сървъра.

Данните се предават като битови потоци от данни.Проблем е предаването на сложни и абстрактни типове данни.RPC предлага решение чрез:

- Marshalling – данните се трансформират до удачен за предаване вариант
- Unmarshalling – обратен процес на marshalling

Свързването е процес, чрез който клиентът създава локална асоциация (handle) за даден сървър при извикване на отдалечена процедура.Има възможност за статично и динамично свързване.

Проблемът с хетерогенността се решава от стъбовете. Възможни решения на проблема с хетерогенността:

- Използване на различни клиентски и сървърни стъбове за всяка възможна комбинация на платформи и програмни езици (2 x n x m).
- IDL – използва се за дефиниране на интерфейси и дефиниране на междинното представяне на данните обменяни между клиента и сървъра.

Асинхронно RPC

Позволява на клиента да изпраща съобщения за заявка на услуга без изчакване на отговор.Стъбът притежава две входни точки за **извикване на процедурата** и за **получаване на резултата.** При липса на резултат клиентът получава съобщение, указващо повторение на запитването. При грешка клиентът получава стойност, показваща характера на грешката.

По същество обработката в **стъбовете е синхронна**, а от гледна точка на **клиента и сървъра** изпълнението е **асинхронно**. Истинска асинхронна обработка е имплементирана в системите управляващи опашки и брокерите на съобщения.

Положителни страни

- Мощна абстракция , скрива детайлите от ниско ниво
- Справя се с хетерогенността

Отрицателни страни

- Изисква доста ресурси (naming , communication, processing, memory (stub and skeleton))
- Проблеми с времето за отговор
- Не е стандарт
- В днешно време е примитив от доста ниско ниво
- Висока свързаност (tight coupling)

Интеграция с използването на Уеб услуги

Дефиниции за уеб услуга

Дефиниция 1

Приложение, достъпно за други приложения през уеб.

Дефиниция 2 (на UDDI консорциум)

Самосъдържащи се, модулни бизнес приложения, които имат отворен, Интернет ориентиран, базиран на стандарти интерфейс.

Дефиниция 3(на W3C)

Софтуерно приложение, идентифициращо се чрез URI, чиито интерфейси е възможно да бъдат дефинирани, описани и открити чрез XML артифакти. Уеб услугата поддържа директно взаимодействие с други софтуерни агенти, използвайки XML базирани съобщения, обменяни посредством Интернет базирани протоколи.

Дефиниция 4

Стандартизиран начин за интеграция на уеб базирани приложения при използване на XML, SOAP, WSDL и UDDI отворените стандарти, основавайки се на Интернет протокол. XML се използва за форматиране на данни, SOAP за тяхното трансфериране, WSDL – за описание на наличните услуги, а UDDI – за откриването им.

19. Интеграция на разпределени информационни системи

Описание на уеб услуги

Основните аспекти в описанието на уеб услуги са:

- Общ мета-език
  - XML се използва като базов език за спецификация на всички езици,необходими за описание на уеб услугите.
- Интерфейси
  - Описанията на уеб услугите са по-обширни (в сравнение, например с CORBA): спецификация на адреси (URI), транспортни протоколи (HTTP).
  - Web Service Definition Language (WSDL) е базиран на XML език , който предоставя модел за описание на уеб услуги.
- Бизнес протоколи
  - Доставчиците на услуги правят опити да наложат правила за комуникация между уеб услугите и клиентите, специфицирани като част от т.нар. бизнес протокол.
  - Съществуват предложения за стандартизиране на езици за дефиниране на бизнес протоколи: Web Service Conversational Language (WSCL),Business Process Execution Language (BPEL).
- Свойства и семантика
  - При избора на уеб услуга освен данни за интерфейса й клиентът има нужда и от допълнителна информация като например нефункционални свойства.
  - Спецификацията на UDDI описва как да бъде организирана информацията за уеб услугите и как да бъдат изградени хранилища за нейното регистриране и заявяване.
- Вертикални стандарти - дефинират специфични интерфейси, протоколи, свойства и семантики,които услугите, предлагани в определен домейн, трябва да притежават.

Откриване на уеб услуги

Описанията на уеб услугите се съхраняват в “директории”,които позволяват:

- регистриране на нови услуги от доставчиците
- търсене на и локализиране на определени услуги от потребителите.

Откриването на уеб услуги може да стане в дизайн режим и режим на изпълнение.(design time and runtime)

Директориите могат да бъдат хоствани и управлявани централизирано и P2P базирано. UDDI спецификацията предоставя стандартни APIs за публикуване и откриване на информация в директориите за услуги.

Протоколи за взаимодействие:

Транспортен протокол

мрежовата комуникация на уеб услугите е скрита зад транспортни протоколи като HTTP.

19. Интеграция на разпределени информационни системи

Протокол за обмяна на съобщения

Simple Object Access Protocol (SOAP) стандартизира начина, по който обменяната информация се форматира и пакетира, т.е. дефинира шаблон за съобщения.

Мета протоколи

- Улесняват и координират прилагането на бизнес протоколите.
- WS-Coordination е спецификация, която прави опит да стандартизира мета-протоколите и начина, по който WSDL и SOAP трябва да бъдат използвани за предаване на информация, релевантна на избрания протокол.

Хоризонтални (middleware) протоколи

- Уеб услугите и поддържащата ги инфраструктура са разпределени, поради което основната middleware комуникация се постига посредством P2P протоколи, наречени хоризонтални протоколи.
- Скрити са от разработчиците и потребителите и се управляват напълно от инфраструктурата.
- WS-Transaction протокола дефинира как да бъдат имплементирани транзакционни свойства (базиран е на WS-Coordination).

Композиция на уеб услугите

- Съставна услуга: уеб услуга имплементирана посредством извикването на други уеб услуги.
- Базова услуга: уеб услуга, достъпваща директно локалната система.

Съставните и базовите услуги трябва да се описват, откриват иизвикват по един и същ начин.

Уеб приложения и технологии за поддръжка на отдалечени клиенти

Аплети

Един от първите решения за пренасяне на динамично съдържание по мрежата са аплетите.Аплетите представляват Java програма, която е вградена в HTML страница. Когата страницата бъде свалена от браузъра, Java виртуалната машина започва да изпълнява кода.

Предимства

- Предимства предоставя част от предимствата на десктоп приложенията
- Предоставя по-голяма динамичност
- Можем да ги използваме за по-сложни клиенти
- Изпълними за всички операционни системи, за които има java виртуална машина.

Недостатъци

- Сигурност – от съображения за сигурност , аплетите се изпълняват в sandbox, с ограничени възможности за взаимодействие с локалната операционна система
- Двоичният java код трябва да се сваля всеки път , когато приложението се достъпи.

19. Интеграция на разпределени информационни системи

Common Gateway Interface

След като аплетите не са подходящи за по-сложни клиенти , следващият подход е да се гледа на уеб сървъра като на интерфейс.Т.е уеб сървърът може да отговори на заявка след като изпълни дадена програма, която ще генерира съответният документ , които да бъде върнат от сървъра.

CGI е стандартен механизъм, които дава възможност на HTTP да си сътрудничи с външни приложения които са като шлюз за локалната информационна система. CGI съпоставя на всяка програма URL , а програмата получава своите параметри чрез параметрите на HTTP заявката.

Предимства:

- Поддържат се различни езици за програмиране
- Поддържат се интерфейси към бази данни

Недостатъци:

- Ако се използва за интеграция на ниво http- няма стандарт , които да преписва синтаксис и семантика на данните (HTML не е задължителен , да не говорим за XML или JSON)
- Поддръжката на много платформи може да е проблем.
- Проблеми с производителността - всяка заявка се изпълнява в совствен процес.Няма спомагателен жизнен цикъл на приложенията.

Сървлети

За да се избегнат проблемите с производителността на CGI, могат да се използват сервлети.Зад тях стои същата идея като за CGI , но има разлика в имплементацията.Първата разлика е , че сървлетите не вървят в собствени процеси , а в собствени нишки. Нещо повече те са част от уеб сървъра, т.е се спестява междупроцесната комуникация в рамките на една и съща операционна система. Тази архитектура позволява допълнителни оптимизации , като имплементирането на кеш, които трудно би риализируем – всяка програми трябва да имплементира свой кеш.

Подобни технологии –ASP, ASP.NET

Предимства

- Имат свой собствен жизнен цикъл, позволяват на приложния програмист да се съсредоточи върхи бизнес логиката.
- Презиползването на инстанциите на сървлетите
- Надграждането с различни технологии
- Всички REST –full услуги в java се базират на Java Servlet Specification
- Могат да се използват за диспачери , филтри

Недостатъци

- Все още имаме примитиви от ниско ниво
- Неудобно е да се използват за писане на Mark-up (HTML).Решение - JSP, JSF

Проблеми:
- файлове
- различни формати
- трудна синхронизация
- конвенции за именуване, правила за заключване, проверки за актуалност
- проблеми за местоположението
- често обмен - неефективно

Но е прост, не се налагат промени в приложенията

- сокети
- трябва да има синхронизация м/у клиента и сървъра (двустранна връзка, подържа се през целия процес на обмен)
- еднакъв формат на данните

- RPC
- изисква доста ресурси
- проблем с времето за отговор (през мрежа)
- не е стандарт (≠ платформи, ≠ езици)

- RMI
- платформено и езиково ~~X~~ зависим

Проблеми:
- файлове
- различни формати
- трудна синхронизация
- конвенции за именуване, правила за запазване, проверки за актуалност
- проблеми за местоположението
- често обмен - неефективно

Но е прост, не се налагат промени в приложенията

- сокети
- трябва да има синхронизация м/у клиента и сървъра (двустранна връзка, подържа се през целия процес на обмен)
- еднакъв формат на данните

- RPC
- изисква доста ресурси
- проблем с времето за отговор (през мрежа)
- не е стандарт (≠ платформи, ≠ езици)

- RMI
- платформено и езиково ~~X~~ зависим