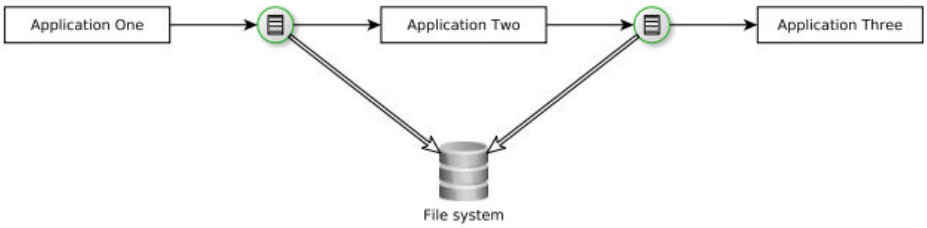
Integrace aplikací, webové služby

- většina aplikací je již distribuovaná, často závisí na jiných systémech

Přístupy – low-level

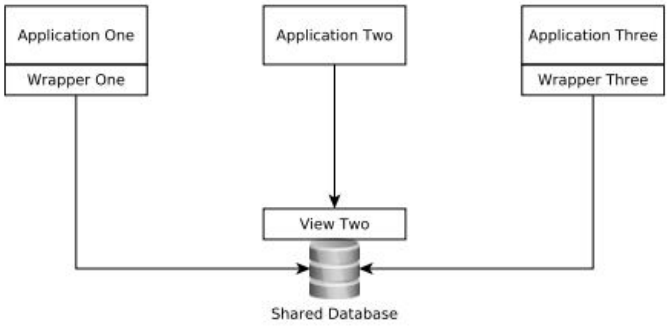
Soubor

- aplikace si vyměňují data zapisováním do sdíleného souboru
- na lokálním systému, pipeline processing
- problém s formátem, schéma, škálovatelností, concurrency,…



Databáze

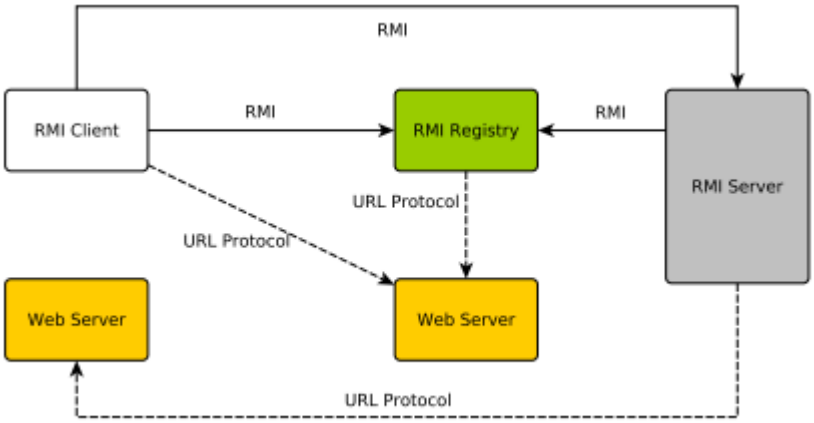
- app sdílí DB, mohou používat různé view na tu samou DB
- není potřeba integrační vrstvy, data jsou vždy aktuální
- problémy: schéma (obecné x komplexní), evoluce schématu, notifikace



Přístupy – platformě závislé

Java RMI

- Remote Method Invocation
- OOP ekvivalent RPC
- klient volá metodu remote interface na lokálním stubu
    - stub = RMI-generovaný proxy objekt reprezentující vzdálenou implementaci
- server implementuje remote interface k exportu metod, které mohou být vzdáleně volány
- RMI registry
    - server se registruje jako poskytovatel vzdálených objektů
    - klient je používá aby hledal vzdálené objekty



Přístupy – platformě nezávislé

RPC (Remote Procedure Call)

- volání subroutine na jiném adresním prostoru (většinou jiný PC)
- client-server architektura
- většinou synchronní
- XML-RPC = standart pro RPC používající XML jako formát zpráv
    - platformě nezávislý, přes HTTP

Request

```
<?xml version="1.0"?>
<methodCall>
   <methodName>examples.getStateName</methodName>
   <params>
      <param>
         <value><int>41</int></value>
      </param>
   </params>
</methodCall>
```

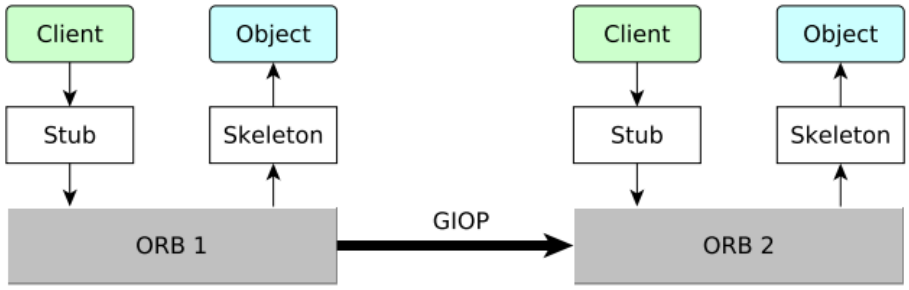
Response

```
<?xml version="1.0"?>
<methodResponse>
   <params>
      <param>
         <value><string>South Dakota</string></value>
      </param>
   </params>
</methodResponse>
```

CORBA - Common Object Request Broker Architecture

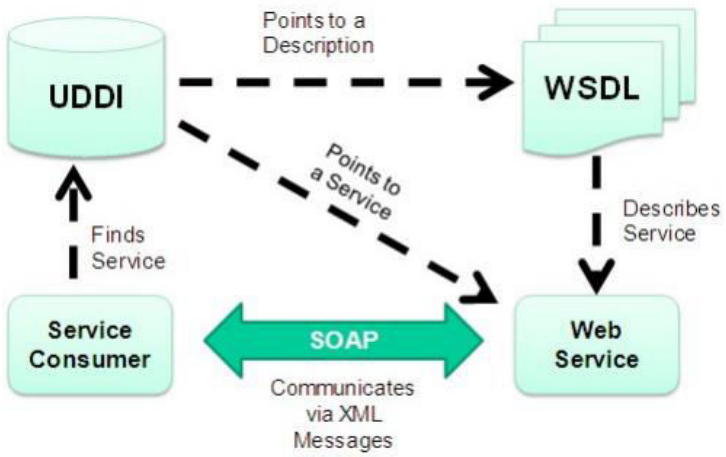
- podobné RPC, ale OO
- klient neví, je-li volání lokální či vzdálené
- má standarty pro definici rozhraní, komunikační protokoly, lokaci,…

- **Interface Definition Language (IDL)**
  - standardizovaný jazyk pro specifikaci rozhraní poskytovanou objektem
  - mapování IDL je ve většina programovacích jazyků
  - používá se pro generování Stubu/Skeletonu
- **Object Request Broker (ORB)**
  - prostředník, umožňuje transparentní lokální a vzdálené volání
  - stará se o (de)serializaci dat, zvládá transakce,…
  - zná lokaci implementace služby
- **General InterORB Protocol – GIOP**
  - protokol pro komunikaci mezi ORBs



SOAP - Simple Object Access Protocol

- standardizovaný protokol pro komunikaci webových služeb
  - WS = SW systém podporující interoperabilní interakci mezi stroji přes síť
- kombo SOAP + WSDL + UDDI
- založený na XML
- oproti CORBA univerzální, XML, bezestavový, může být asynchronní
- zpráva se skládá z:
  - **envelope** (obálka) – jedna na request/response
  - [hlavičky] – doplňující informace (timeout,…)
  - **body** – data
  - [fault] – zpracování chyb
- přes HTTP POST
- vždy klient-server model interakce
- komplexní struktura



- **WSDL = Web Service Description Language**
  - XML popis rozhraní webové služby
  - klient podle něj pozná, jak komunikovat s danou službou
  - nepotřebuje pak generovat skeleton ani stub

- **UDDI = Universal Description, Discovery and Integration**
  - univerzální definice WSDL popisů SOAP webových služeb
  - zjednodušuje objevování WS

Architektury

- vertikální rozdělení = rozdělení logických úrovní systému
- horizontální rozdělení = rozdělení klientů a serverů
- dočasné rozdělení = dle synchronní/asynchronní komunikace,…

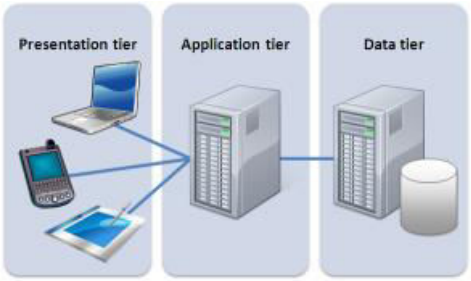
Client-Server vs. Distributed Objects

- **klient-server:**
  - je k nim přistupováno jinak
  - servery zpracovávají požadavky, nabízejí funkcionalitu
  - klient vytváří požadavky, konzumují funkcionalitu
  - SOAP, REST, HTTP,…

- **distribuované objekty:**
  - objekty jsou ekvivalentní, mohou se volat navzájem
  - Java RMI, CORBA,…

Vertikální rozdělení

- vrstvy jsou rozděleny mezi procesy, mohou být i mezi stroji
- **single-tier** = terminal/mainframe configuration
- **two-tier** = client + server
- **three-tier** = typické, separátní klient, server application a databáze



Services (služby)

- **Service Oriented Architecture (SOA)**
  - systém je rozdělen na samostatné jednotky – služby
  - služby si navzájem poskytují funkcionalitu
  - mohou být vyvíjeny odděleně, používat různé technologie, být odstraněny nebo nahrazeny bez vlivu na systém jako celek
  - není to stejné jako Web Services!
  - příklad: SSO, text analysis service

- **Microservices**
  - není přesná definice, pro někoho je to pokročilější implentace SOA / podmnožina
  - SW jednotky komunikují přes jednoduché mechanismy (HTTP)
  - mohou mít vlastní DB
  - mohou být nasazeny nezávisle na ostatních

- komunikace v SOA:
  - **Enterprise Service Bus (ESB)**
    - podniková sběrnice služeb
    - prostředník
    - spojuje a zprostředkovává komunikace/interakce mezi službami
    - lze je rychle měnit, připojovat, řídit,…
    - mohou podporovat vícero protokolů – SOAP, REST,…
    - jednoduché (Apache Kafka,…) či pokročilé (Oracle, IBM,…)

  - **Smart Services and Dumb Pipes**
    - microservices – decentralizované řízení, často P2P
    - designový princip upřednostňující základní, časem otestované, asynchronní mechanismy komunikace před komplexními

Peer to Peer (P2P)

- decentralizovaná architektura
- nody fungují jako servery a klienti
- pro distribuci obsahu, sdílení, grid computing
- různé typy:
  - **nestrukturované**
    - žádný centrální node
    - peery se navzájem objevují (začnou s malým počtem možných spojení, pak se seznam dalších peerů rozšiřuje)
  - **strukturované**
    - síť má topologii, je efektivnější na objevování
  - **hybridní**
    - kombinace P2P a klient-server
    - server obvykle pomáhá klientům s objevem peerů, vyhledáváním apod.