
N(=3)-Tiered Client-Server Architecture

Jing Dong

CS6362 Software Architecture and Design

1

2-tiered Client-Server Styles

- What is 2-tiered client-server style
 - client is user interface, service is usually database, separated by network
 - synchronous (remote procedure call) message passing
 - Server's role
 - Binds to an Internet (IP) address
 - Listens for clients on the network
 - Provides services to clients that connect to it
 - Client's role
 - Knows the server's IP address
 - Connects to the server
 - Sends service requests

2

2-tiered Client-Server Styles

- Problems with 2-tiered client-server style
 - usability — expose complexities of distributed environment
 - performance — limited by network
 - interoperability — often limited to DBMS
 - Application writer must be able to debug network-level issues and problems
 - Mixes application logic with networking logic
 - If client and server run on different byte-ordered machines, they must be aware of these differences in talking to each other

3

Example Business Rule

- $\text{pay} = \text{hours_worked} * \text{pay_rate}$
- In a client/server architecture:
 - Prompt the user for **employee_number** & **hours_worked**
 - Fetch **pay_rate** from db
 - `select pay_rate from pay_table where employee_id = <id>`
 - Calculate the pay for the employee
 - Generate and execute an SQL statement to update the db
 - `update payroll
set pay = <calculated_pay>
where employee_id = <id>`

4

Change to a Business Rule

- Suppose you need to change the system to account for overtime

```
If (hours_worked < 40)
    pay = hoursWorked * payRate;
else {
    pay = 40 * payRate;
    overtimeRate = payRate * 1.5;
    overtimeHours = hours_worked - 40;
    pay += overtimeHours * overtimeRate;
}
return pay;
```

- Multiple client program needs to be modified, re-compiled, re-tested, and re-installed.

5

Alternately

- A database stored procedure could be used to compute the pay. e.g.,
 - Oracle PL/SQL
 - Java extension to db
- Clients could then concentrate exclusively on presentation.
- Single database would have to be changed, re-tested & migrated.

6

Basic Problems with this Approach

- Want to change the db as little as possible.
 - the most fragile component
- DB is not a great execution engine
 - inefficient
 - limited choice of language
 - hard to interact with outside services
 - poor development environment
 - poor error recovery
- Vendor lock-in

7

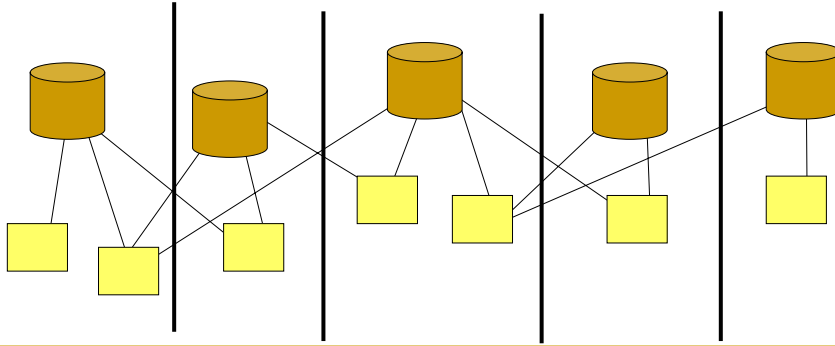
Architectural Problems

- Client-resident business rules
 - client bloat + lack of scalability on client machines
 - need to address lowest common denominator machine
 - 386 with 16M
 - transactions involving more than just db (e.g., queues)
 - must configure all client machines!
- DB-resident business rules
 - db bloat (too much for the db to do)
- Common Issues
 - large # db connections
 - lack of support for caching
 - wide-area data distribution (data partitioning strategy)
 - fault tolerance

8

Legacy Issues

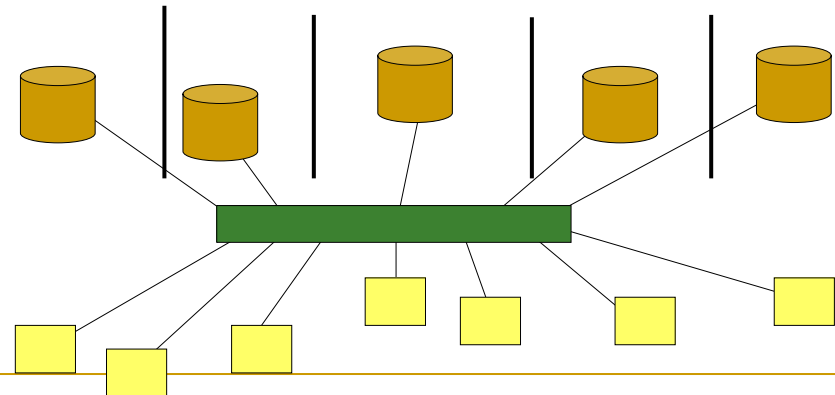
- In large corporations, different departments develop their own client/server systems
- Inevitable in the case of mergers and acquisitions



9

Solution

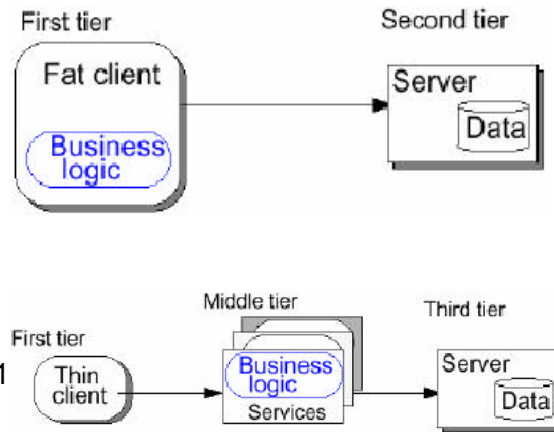
- Add a middle tier to isolate clients from databases.
- Re-engineer the databases going forward.



10

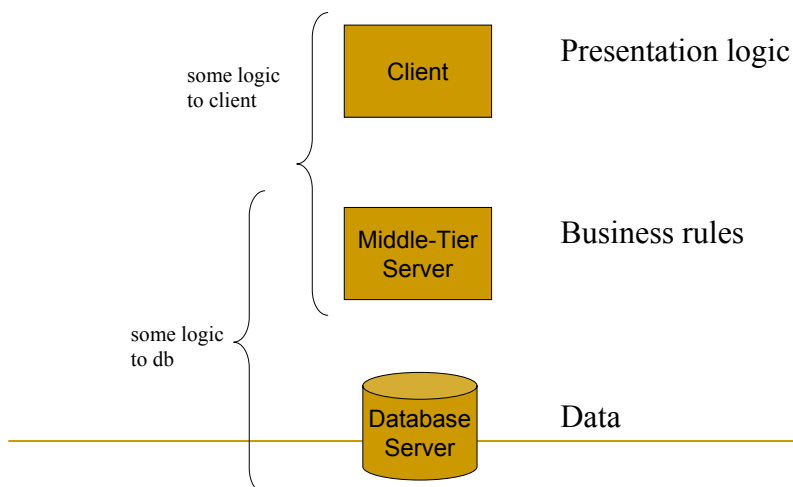
System Architecture Choices

- Monolithic
 - 1 large program, imports/exports data
- Client/Server
 - collection of clients, updates database
 - “fat client”
- 3-tiered
 - collection of clients, 1 mid-tier process for “business rules”
 - “thin client”



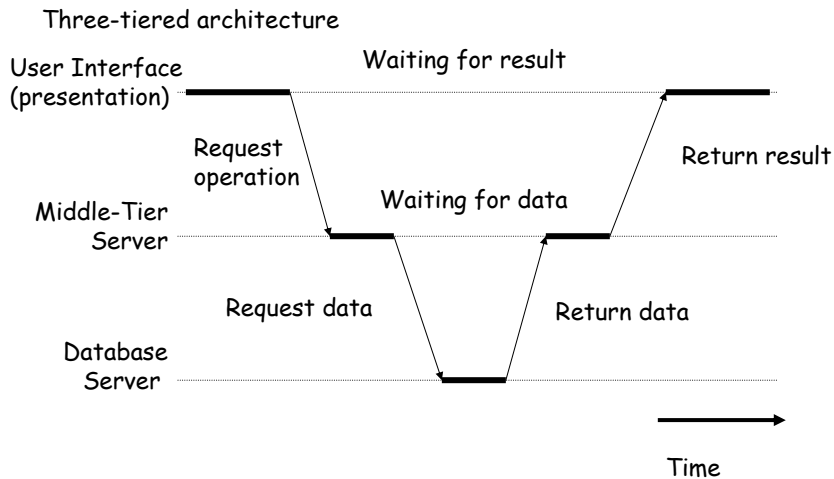
11

3-Tiered Systems



12

3-Tiered Systems



13

Middleware

- Why is it called “middleware”?
 - It's right in the middle of a client and a server
 - Hides operating system and low-level details from the application developer
 - Also called “distributed object computing”
 - CORBA, EJB, DCOM
- Why do we need/have middleware?
 - It makes it easier to write distributed applications
 - Takes care of all the networking code and the messaging
 - Leaves you free to focus on writing the application

14

Where Is Middleware Used?

- Electronic commerce (e-commerce) and electronic trading
 - Goods and services are just as important to businesses as information
 - Competition implies that businesses strive for better ways to manage, present and exchange information, goods and services
 - Businesses automate and computerize their services for more efficiency
- Business-to-Consumer (B2C)
 - Human involved in the communication with the computers of a company
- Business-to-Business (B2B)
 - Computers of one company talking to those of another company
- Diverse kinds of enterprise applications
 - Server farms
 - Finance & banking
 - Supply-chain (inventory) management
 - Enterprise resource planning
 - Storage area networks

15

Technologies In The Different Tiers

- **Front-end clients**
 - Graphical user interfaces
 - Web-based clients
 - CORBA (Common Object Request Broker Architecture) clients
 - J2EE (Java 2 Enterprise Edition) clients
 - DCOM (Distributed Component Object Model) clients
- **Middle-tier servers** (business logic)
 - Web servers
 - CORBA components/servers
 - J2EE servers
 - DCOM servers
- **Back-end systems** (usually existing technologies or products)
 - Databases (e.g., Oracle)
 - Enterprise messaging systems (e.g., IBM's MQSeries, TIBCO's Rendezvous)
 - Email systems
 - Enterprise naming and directory services (e.g., LDAP –Lightweight Directory Access)

16

Let Front-End Talk To The Middle-Tier

- There are different ways in which the front-end clients can talk to the middle-tier servers, depending on the middleware you use
 - HTML (Hyper Text Markup Language)
 - Set of rules for publishing text on the World Wide Web
 - HTTP (Hyper Text Transfer Protocol) & HTTP/S
 - TCP/IP protocol used for sending all web information across the Internet
 - XML (Extensible Markup Language)
 - Set of rules describing how you can format and exchange data
 - IIOP (Internet Inter-ORB Protocol)
 - TCP/IP protocol used for communication between CORBA clients and servers
 - SOAP (Simple Objects Access Protocol)
 - Set of rules for peer-to-peer distributed communication over XML
 - SSL (Secure Sockets Layer)
 - Enterprise Security

17

Let Middle-Tier Talk To The Back-End

- There are different ways in which the middle-tier servers can talk to the back-end systems, depending on what the back-end system does
 - JNDI (Java Naming and Directory Interface)
 - Accesses information in enterprise naming and directory services
 - JDBC (Java DataBase Connectivity)
 - Accesses relational data from Java
 - JTA (Java Transaction API)
 - Manages/coordinates transactions across heterogeneous enterprise systems
 - JMS (Java Message Service)
 - Sends/receives messages over enterprise message systems
 - Java Mail
 - Accesses existing email systems

18

Challenges For Enterprise Applications

- Too many enterprise technologies!
 - A new technology is born every few months
 - No single accepted technology for building applications
 - There is no “one ring to bind them all”
 - *Strategies*: IIOP was one attempt, and can glue EJB and CORBA together, but what about the rest?
- Integration with legacy systems
 - There’s millions of lines of code already written and a lot of time and \$\$ invested on using older technologies
 - How do new technologies cope with integrating legacy code?
 - *Strategies*: “wrappers” around legacy code to make it easier for it talk to new code

19

Challenges For Enterprise Applications

- Dependability
 - Enterprise applications deal with \$\$, and customers hate losing \$\$
 - Need to provide services 24x7 even if computers and software fail
 - *Strategies*: replication, fault detection, recovery, check pointing
- Security
 - A lot of enterprise applications deal with \$\$ and private information (social security numbers, credit card numbers, mother’s maiden name, etc.)
 - How do you know you can trust these systems, and not resorting to burying your savings in your backyard?
 - *Strategies*: firewalls, encrypted communication

20

Challenges For Enterprise Applications

- Latency & Real-Time
 - Responsiveness is crucial
 - *Strategies*: locality-based access, and a lot of proprietary practices which often break and perform poorly under fault & load conditions
- Handling load and congestion
 - Accessing services under congestion and peak load (e.g., CNN.com on 9/11)
 - Need for ways to cope with multiple clients simultaneously
 - *Strategies*: load balancing, simultaneous execution of multiple requests (using multithreading), etc.

21

Some Industry Statistics

- 2/3 of respondents had a formal system architecture
 - Monolithic
 - 14%
 - client/server
 - 26%
 - n-tier client/server
 - 54%
 - web centric
 - 3%
- Source
 - Cutter Consortium
 - Jan, 1999
 - survey of Fortune 1000 internal IT projects
 - "Client-server in general, and n-tier client-server in particular, gives IT the flexibility to deploy available computing resources most effectively."

22

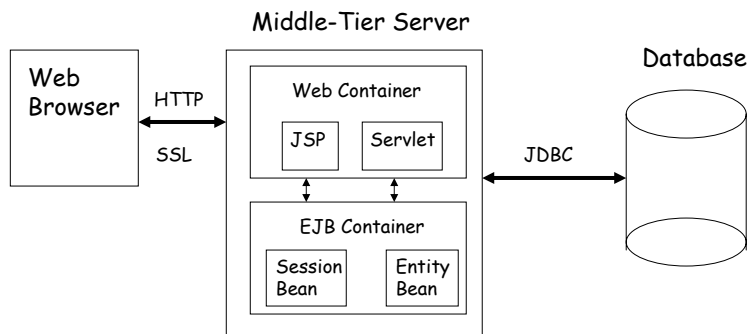
J2EE (Java 2 Enterprise Edition)

- Sun Microsystems' Java-based middleware
 - Implicitly has all of Java's portability advantages
 - Component: application-level software unit
 - Container: runtime support/services for a component
- On the client side
 - Applets, application clients executing in their own JVM
- On the server side
 - Web components
 - Servlets and Java Server Pages (JSP)
 - Enterprise Java Beans (EJB) components can contain two kinds of objects
 - Session beans: valid for the duration of a client-server session
 - Entity beans: valid for the lifetime of the persistent data that it manages
 - What makes J2EE powerful?
 - Can interface to multiple existing back-end systems and technologies
 - Power of Java's intrinsic portability, graphical & garbage collection capabilities

23

J2EE

J2EE Platform Architecture



24