Understanding Multi-tier Client/Server Systems in a Composer Environment

Session 430

Jeff Chancellor
Texas Instruments

Contents

- Some background on client/server architectures
- Prevalent styles in use today
- Technical implementations
- Existing Composer practices
- Scenario details and project style criteria
- Future design considerations using Composer and Arranger

Background Information

- Definition of client/server terminology
- Prevalent styles of client/server systems
  - Two-tier styles
  - Three-tier styles
  - Variations of the tiered styles
- Installed base of these styles
- The multiple tier dilemma

Client/Server Versus Distributed Computing

- *Client/Server Computing*
  Client/Server computing involves two separate programs, whether on the same platform or different platforms, that act in specific roles as client or server.
- *Distributed Computing*
  Distributed computing involves two or more programs in which the roles are not specific.

Client/Server Versus Network File Systems

- *Client/Server Computing*
  Client/Server involves programs that cooperate in order to satisfy an applications functionality.
- *Network File Systems*
  Services typically are oriented around file sharing, print sharing, or extended disk access. There is no real applications work occurring.

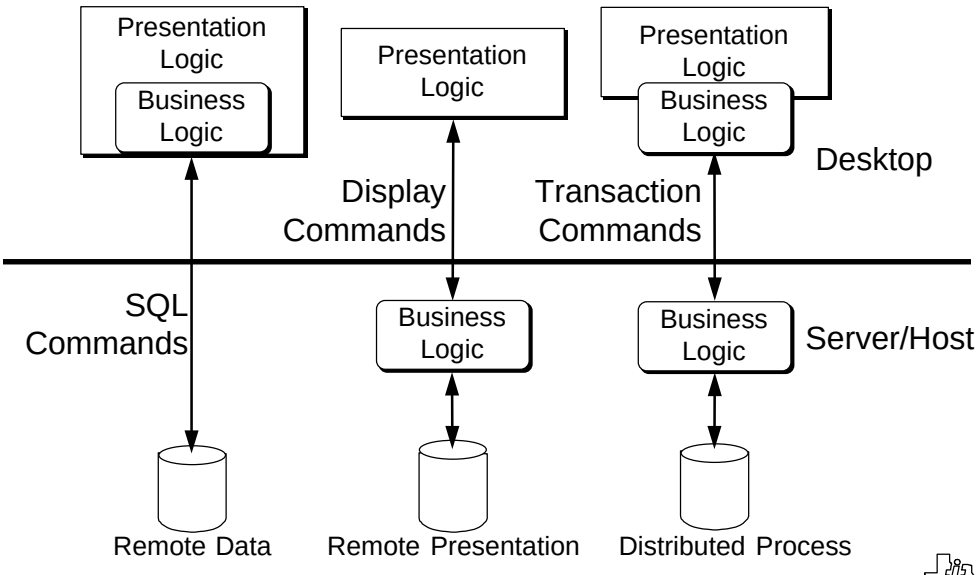
Characteristics of Client/Server Computing

- *Communication facilities and structure*
  In most of the installed base of client/server systems today, the communications are based on TCP/IP and the nature of the conversation is synchronous.
- *Dedicated roles as client or server*
  Whenever a client/server conversation is initiated, the client simply waits for the server to respond. The client always plays the role of the requester, while the server is always the responder.

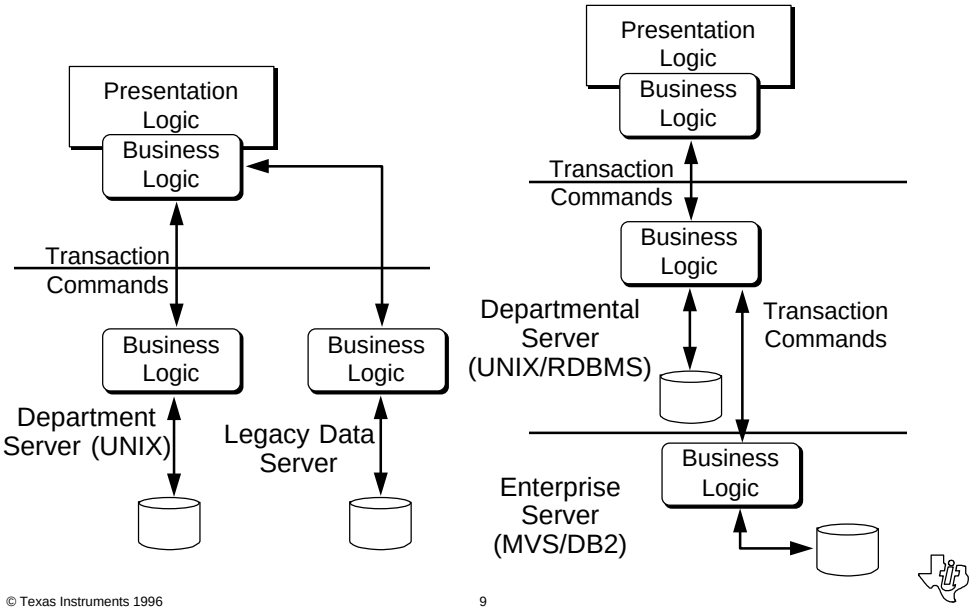
Characteristics of Distributed Computing

- *Communication facilities and structure*
  In a distributed environment, the communications are typically based on TCP/IP and are either synchronous or asynchronous.
- *No definitive roles as either client or server*
  In an asynchronous environment, both the client and server must be able to respond to unsolicited requests.

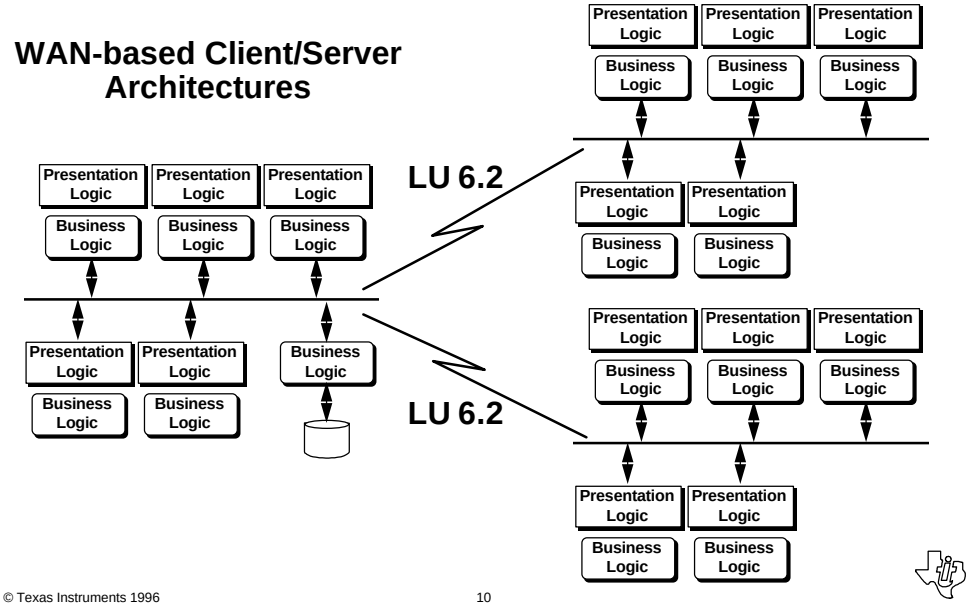
Client/Server Styles



More Client/Server Styles



Even More Architectures



Newest Member of Client/Server Styles

- Show a picture of the Internet Client/Server Model
  - Thin Client
    - » Applets
    - » Browsers
  - Web Server/Communications Gateway Interface
  - Transaction/Data Server

Styles in Use Today

- *Remote Data Access*
  It is estimated that 80% of the existing client/server applications use a Remote Data Access style of computing. These applications use data access and communications facilities provided by the DBMS vendor.

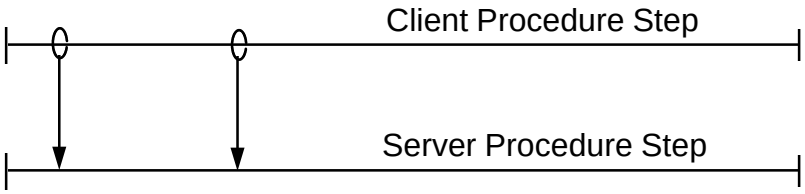
Styles in Use Today

- *Distributed Process*
  Less than 20% of the existing client/server applications are distributed across platforms. Of these applications, the applications typically use a custom-developed communications facility. Of these distributed systems, the Application Server style is most often used as the distribution mechanism.

Composer Practices

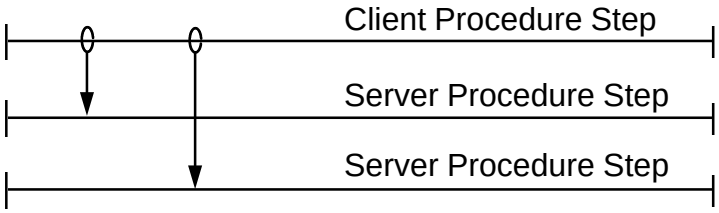
- Dialog Design Diagram (DLG) is the primary facility for creating:
  - Client procedures
  - Server procedures
  - Transaction request between client and server
  - Conversation request between clients

Two-Tier Client/Server Composer Style



- Separate procedures promote procedure reuse
- Concept of public versus private servers
- Link flow represents the transaction request

Three-Tier Client/Server Composer Style



- Client initiates flows to each server
- Servers can be homogeneous or heterogeneous platforms

The Multiple-Tier Dilemma

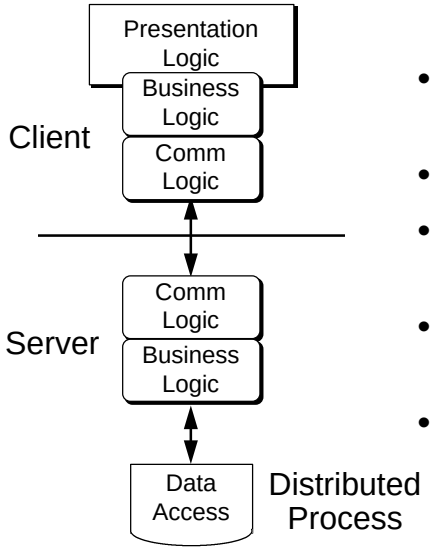
- Why are multiple-tier systems being considered
  - Elegance instead of Simplicity
  - Flexibility instead of Rigidity
  - Performance instead of performance????

Technical Implementations

- Two-Tier Transaction Processing
- Three-Tier Transaction Processing
- Two-Tier Distributed Data
- Multiple-Tier Distributed Environment

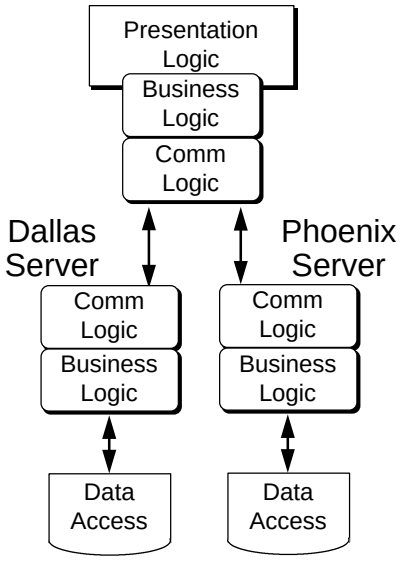
In this presentation, a "TIER" will be defined as a specific computing platform that is executing some part of the application (e.g., Presentation, Business Rules, Communication, or Data Access)

Two-Tier Environment



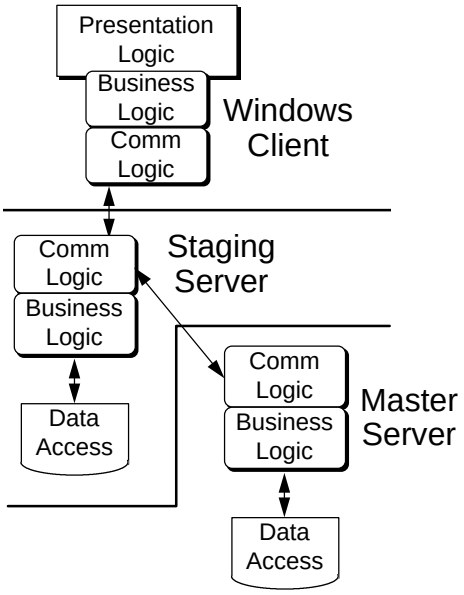
- Presentation allocated to client
- Business rules split
- Communications initiated by client
- Data access and common routines allocated to server
- Printing routines typically handled by client

Two-Tier Distributed Data



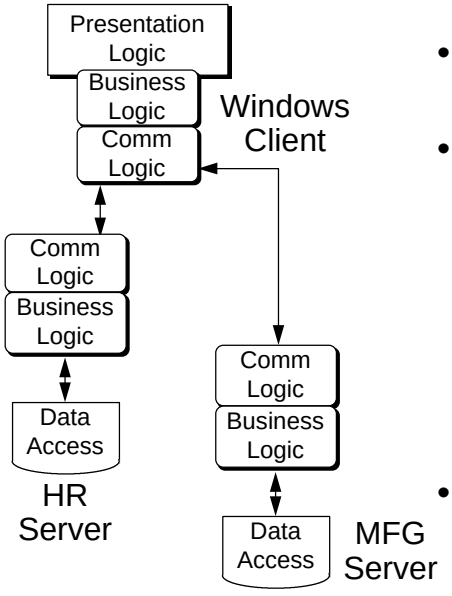
- Database structure replicated
  - Records owned by functional organization or location
  - Roll-up processing for “Master Records”
  - Roll-up interval defined
- Distribution greatly enhances performance (reduced storage at each server)
- Same DBMS used at each server location

Three-Tier Transaction Processing



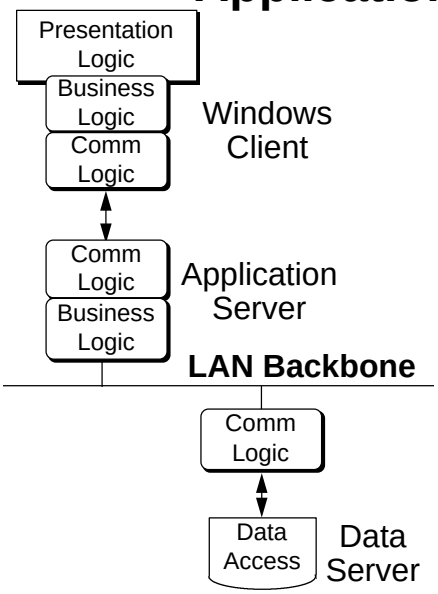
- Transaction “Staging”
- Business logic on the staging machine may/may not require data to validate
  - Reduces “Master Server” processing requirement
  - Used where number of incorrect transactions are high

Multi-Distributed Environment



- Implemented in large, autonomous organizations
- Functional areas have their own data centers
  - Responsible for all data and server processing
  - May interconnect as required by the business
  - Little replicated data
- True heterogeneous network

Application Server Style



- Fast execution on application server platform
  - High-speed RISC server
  - High-speed connection to data sources
- Participates in network monitoring program (e.g., OpenView)
- Multiple middleware support

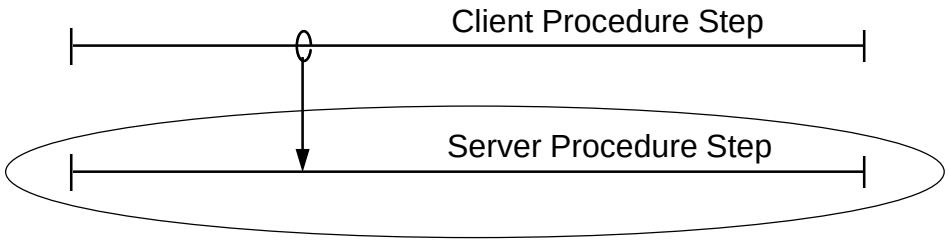
Layering for Application Servers

- Separate server-side business logic from the data access component
- Isolate the communication decision processes from the business logic
- Identify the transaction control points in the application server procedures

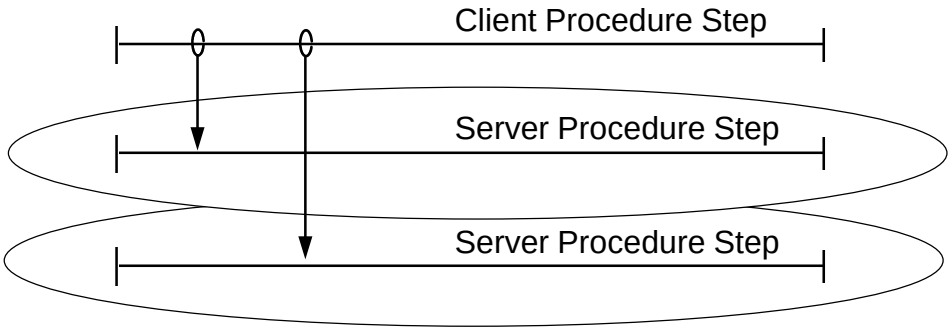
Transaction Terminology

- Definition of a transaction
- The transaction envelope
- Transaction control for commits/rollbacks

The Transaction Envelope
Two-Tier Application



The Transaction Envelope Multi-Tier Application



Transaction Control

- Transaction control resides on client in Remote Data Access applications
- Transaction control resides on server in Distributed Process applications
  - Each server in multi-tier applications
  - Application server in such architectures
  - Initial server in “staging” applications

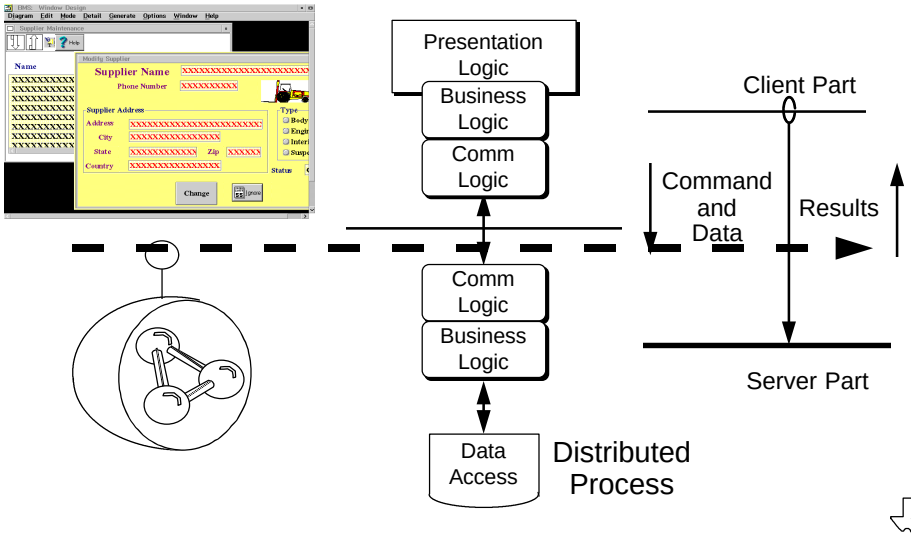
Defining the Client/Server Architecture

- Planning steps
- Architecture steps/investigations
- Criteria for choosing the right style

Design Criteria for Architectures of Client/Server Applications

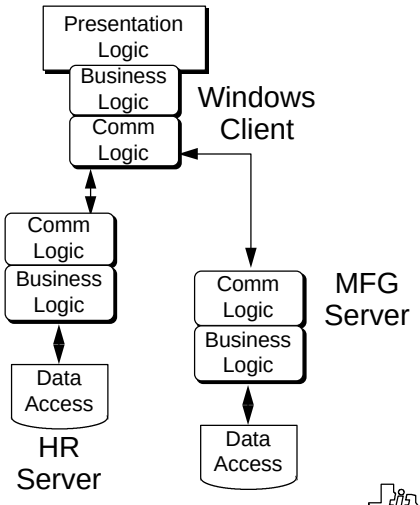
- What is the optimum architecture for client/server applications
- Performance
  - LAN/WAN throughput
  - Client platform specifications
- DBMS structure and distribution strategy
- Common routines
  - Security
  - Access control
  - Common processing/application integration

Mapping Design Objects to Technical Implementations



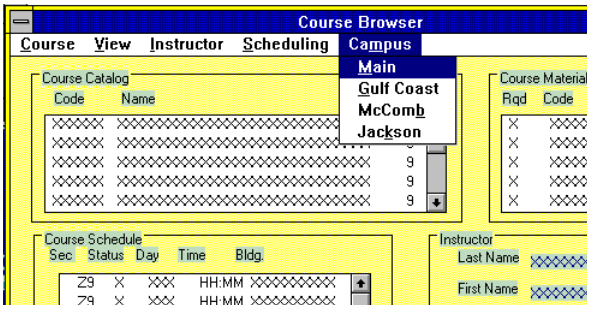
Multiple Tier Scenario Existing Composer Approach

- Application needs data from both HR server and MFG server
  - Initiates flow from client to HR server first
  - Once HR data is received, then MFG data is retrieved
- Result is to calculate BONUS of factory employee



GUI Design Suggestions for Distributed Data Servers

- Add MENUBAR pulldown for Servers
- Use MARK or UNMARK to show location selected
- Use DROPDOWN listbox for Servers



Errors and Recovery Concepts

- Detecting and reporting error conditions
- Recovering from error conditions
- Distinguishing from errors and “invalid business” conditions/constraints
- Keeping track of processing

Summary and Conclusions

- Composer 3 supports multi-tier client/server applications
- Carefully match the application to the appropriate style of processing
- Understand the network capacities and bandwidths required by the application
- Match distributed data strategies to access techniques

Understanding Multi-Tier Client/Server Systems in a Composer Environment

Session 430

Jeff Chancellor

Texas Instruments