11. Application Servers
(a.k.a. TP Monitors)

CSEP 545 Transaction Processing
Philip A. Bernstein

Copyright © 2005 Philip A. Bernstein

11.1 Introduction

An application server coordinates the flow of requests between message sources (displays, applications, etc.) and application programs that run requests as transactions.

Application Server Components

- **Web Browser**
  - A smart device, with forms, menus, input validation
  - Calls the web page associated with the URL, which in turn calls a workflow controller

- **Workflow Controller**
  - Calls Start, Commit, and Abort
  - App logic that transforms request (automatic loan payment, money transfer) into calls on basic objects (loan, account).
  - Sometimes called business rules.

- **Transaction Server**
  - Business objects (customer, account, loan, teller)

- **DBMS**
  - Database Management System

Application Server Functions

- **Glue and veneer for TP applications**
  - Glue fills in gaps in system functionality
  - Covers the interface with a seamless veneer

- **Natively, it provides run-time functions for applications**
  - (workflow controller and transaction servers)

- **OS functions**
  - Threading and inter-process communication, often passed through from the underlying OS

- **Dist'd system functions**
  - Transactions, security, queuing, naming, service, object pools, load balancing...

- **Portal functions**
  - Shopping cart, catalog management, personalization...

Provides some application development tools
Provides system mgmt for the running application.

Application Server Products

- **Apple WebObjects**
- **BEA Tuxedo**
- **BEA W eblogic**
- **IBM CICS**
- **IBM IMS/DC**
- **IBM WebSphere**
- **Iona I Portal App Server**
- **Java J 2EE**
- **Lotus Domino**
- **Matsushita E i HP Enterprise Services**
- **MS Transaction Server**
- **MS SQL Server**
- **Novell S erves**
- **Oracle Application Server**
- **Oracle E Business Suite**
- **Sybase Enterprise Server**
- **Sun ONE Application Server**
- **Sybase EA Server**
- **And many others. See serverwatch.internet.com**
11.2 Two-Tier vs. Three-Tier

Before the web, most all-to-end user scale apps were in plan ented in 2 tiers on a LAN
- PC runs a 4GL, such as Visual Basic, Powerbuilder, or Delphi
- Server system includes transaction server, application and DBMS

Two-Tier for the Web

Presentation server ⇒ Web server
- In essence, the web browser is a device
- Web server invokes a web page that has embedded script
  - Active Server Page (ASP.NET) or Java Server Page (JSP)
  - Page file extension tells the web server to run the ASP/JSP interpreter
  - Script can include DBMS calls and can run as a transaction

Two-Tier is Enabled by DBMS Stored Procedures

- Stored procedure - An application procedure that runs inside the DBMS
  - Often in a proprietary language, such as PL/SQL (Oracle), T-SQL (MS, Sybase)
  - Moving to new languages, such as Java
- In plan ent transaction servers as stored procedures
- Use DBMS client-server protocol
- No application server needed
  - Hence, som etimes called "TP lite"

Scalability Problem of Two-Tier

- 2-tier is feasible, but does not scale as well as 3-tier due to session management
  - Session - shared state between communicating parties
  - Entails a memory cost and a setup (processing) cost
  - Sessions reduce amount of per-request context passing (com. addresses, authenticated user/device)
  - Standard DB APIs (e.g., ODBC) work this way
  - Hence, in 2-tier, N clients and M servers ⇒ N·M sessions
    - E.g. 105 presentation servers and 100 servers ⇒ 107 sessions
- Partition presentation servers across workflow controllers
  - Each workflow controller still connects to all transaction servers but there are many fewer workflow controllers than presentation servers

3-Tier Reduces the Number of Sessions

- Partition the set of presentation devices (e.g., 100 PS per WF C)
- 100 W F C · 100 PS per FC · 103 Presentation servers = 1,000,000 sessions
- Typo on page 50 in textbook, paragraph 4: the web server could have 10,000 sessions instead of 150,000 sessions

2-Tier vs. 3-Tier - Other Issues

- In early 90's people argued whether 2-Tier was enough
  - Scalability was the decisive factor, but there were other issues
- Database Servers
  - N-standnored DBMS language, usually less expressive
  - Without decisional tooling and another language to learn
  - Limited interoperability of cross-server calls
  - Limited interoperability of distributed transactions
- Poor flow in OO design, which are inherently 3-tier
  (client, business rules, business objects)
- Application Servers
  - More system complexity
How the Web Changed Things

- Presentation server => Web server

All requests have to pass through a Web server
- In 2-tier, each Web server needs sessions to all DB servers
- Session reduction by workflow control is less critical but still useful.

Workflow control is still useful for request management
- Calling Start, Commit, and Abort
- Encapsulating business rules that transform requests into calls on basic objects

11.3 Web Servers

- Presentation independence - application independent of the display device used
  - Today, this is via http and https
  - In the past, it was via a display controller or middle-tier
    - UNIX puts where presentation functions insulated the rest of the back-end system from different device types
- Web server performs presentation functions:
  - Gathering input
  - DB caching
  - Validating input
  - Authentication
- They also do some basic request routing
  - Constructing requests
  - Invoking applications
- Examples - IIS (MS), Apache, Netscape Server

Gathering Input

- Gathering input - Select transaction type (menu item, etc.), and fill in a form (request's parameters)
  - Today, Web forms, moving to XML (XForms, XSLT, ...)

30 year evolution of presentation devices
- Teletype, character-at-a-time terminal (async), block-mode terminal (IBM 3270)
- Specialized devices - ATM, barcode readers, gas pumps, robots, credit card authorization, cash registers, printers, etc.
- 4GL on a PC - ActiveX controls accessed from Visual Basic (VB), Pow erBuilder, Delphi, etc.

Caching

- Every process-to-process call has a cost
  - Adds to response time and consumes resources
- Use a cache in Web server to avoid calling workflow controller or DB system
  - Cache popular read-only data that need not be refreshed frequently
    - E.g., catalog items, sale items, cover page at an auction site, recent news, etc.
  - Also, data required for input validation info

Input Validation

- Validate input against locally cached tables
  - E.g., product types, departments, numbers
- A workflow setting com inunications and server resources for obvious input errors
  - Fewer round-trips to the DBMS
  - And faster feedback to the end user
- "Cache" in part of the Web page
  - Listboxes, scripts
  - Cache size is a factor (it affects page access time)

Authentication

- Authentication - determining the identity of a user and/or display device
  - Client system (e.g., PC) may do authentication, but the server usually does it too (doesn't trust clients)
  - Encrypt the wire to avoid wiretapping and spoofing (in the Web, use https - http over secure socket layer)
- Geographical entities - check that a particular device is allowed access (e.g., security trading room)
- Need system management functions to create accounts, initialize passwords, maintain hours of access (simplify it using a role abstraction)
Constructing Requests

- A request includes:
  - User id – for authorization and personalization
  - Device id – where to send a reply
  - Device type – what message types can it understand?
  - ObjectID – in a O O setting
  - RequestID – to ask later about request status & to link a reply
  - Request type – name of transaction type requested
  - Request specific parameters

- Can be combined with protocol header (e.g., http header)

Application Invocation

- Request arrives as an http message.
  - Need to call a program (i.e., WF C), to perform the request
  - Common Gateway Interface
    - Write a script, store it as a file in cgi-bin
    - Web server creates a process to execute the request (Slow!!)
    - ISAPI (Microsoft) and NSAPI (Netscape)
      - Web server calls an in-proc .dll instead of creating a process
      - Web server can cache the .dll
      - More complex programming model, but much faster
  - Active Server Pages and Java Server Pages
    - Offers the performance of ISAPI with programmability of CGI

Load Balancing

- Web servers enable scale out, so you can just add more server boxes to handle more load.

- To solve this problem:
  - Ensure all web servers are identical (no server-specific state)
    ⇒ don't retain client state on web servers (hard to avoid...)
  - Randomly assign requests to servers (e.g., use an IP sprayer)
  - Avoid sending requests to a failed server

11.4 Transaction Bracketing

- For the most part, Workflow Controllers (WF C) and Transaction Servers are just plain old server programs

- The main WF C differentiating features
  - Brackets transactions (issues Start, Commit, and Abort)
  - Handles Abort (returns cause of the Abort)
  - Does not access the DBMS

Nested Transaction Calls

- What does Start do, when executed within a txn?
  1. it starts an independent transaction, or
  2. it does nothing, or
  3. it increments a nested transaction count (which is decremented by each commit and abort), or
  4. it starts a sub-transaction

- (2) and (3) are common.

- Enables a transaction bracketed program to be called by another transaction

- (1) implies Be Careful!

Transparent Transaction Bracketing

- Transaction-hood is a property of the app component.

- In COM+, a class is declared:
  - Requires new – caller a component starts a new transaction
  - Required – if caller is in a transaction, then run callee in caller’s transaction, else start a new transaction
  - Supported – if caller is in a transaction, then run callee in caller’s transaction, else run outside of any transaction
  - Not supported – don’t run in a transaction

- Caller can create a transaction context, which supports Commit and Abort (chained model)
  - Caller issues SetCompletewhen it’s done and willing to commit, or SetAbortto abort.
Transparent Txn Bracketing (cont'd)

- EJB and J2EE
  - Implements COM+ technology in Java: Requires New, Required, Supported, NotSupported
  - It came later, so there are two additions.
  - Mandatory: If caller is in a transaction, then run the callee in that transaction, else raise an exception
  - Never: If caller is in a transaction, then raise an exception

Runtime Library Support

- TP services require runtime library support
  - May or may not be language-specific
  - Language-specific:
    - Java 2 Enterprise Edition (J2EE, formerly Enterprise Java Beans)
      - Encapsulates runtime library as a container object.
      - BEAW eblogic, IBM WebSphere, ...
    - Older examples are Tandem Pathway (Screen COBOL) and Digital's ACMSxp (Structured Txn Lang)
  - Language-independent runtime library
    - MS COM+, IBM's CICS, Oracle App Server, ...

Savepoints

- Savepoint - a point in a program where an application saves all its recoverable state
- Can restore a savepoint within the transaction that issued the savepoint. (It's a partial rollback.)
- SQL DBMSs use them to support statements in SQL statements.
  
  ```
  Start;
  get-request;
  Savepoint("B"); ...;
  if (error) {Restore("B"); ...; Commit;} 
  ...;
  Commit;
  
  Savepoints are not recoverable. If the system fails or the transaction aborts, the transaction is completely undone.
  ```

11.5 Processes and Threads

- Application Server architecture is greatly affected by
  - which components share an address space
  - how many control threads per address space
- TP grew up in the days of batch processing, and reached maturity in the days of time sharing.
- TP users learned early that a process-per-user fails:
  - Too much context switching
  - Too much fixed memory overhead per process
  - Process per user per machine, when distributed
  - Some OS functions scan the list of processes
  - Load control is hard

Multithreading

- Have multiple threads of control in an address space
- Used to be a major Application Server feature
  - Application Server installs threads when app calls Application Server function that blocks
- Now, most OS's support it natively
  - Can run a process's threads on different processors (SMP)
- Whether at the user or OS level,
  - multithreading has few resources and less context switching
  - but little protection between threads and a server failure affects many transactions

Mapping Servers to Processes

- Presentation, web servers, workflow controllers, and transaction servers are multithreaded servers
- Costs 1500 - 25,000 instructions per process call, vs. 50 instructions per local procedure call...
  - but it scales with flexible configuration and control
### 11.6 Remote Procedure Call

- **Remote Procedure Call (RPC)**
  - Program calls remote procedure the same way it would call a local procedure.
  - Hides certain underlying complexities
    - Communication and message ordering errors
    - Data representation differences between programs

#### Transactional RPC
- Ideally, starts returns a transaction ID that's hidden from the caller.
- Procedures don't need to explicitly pass transaction ID's.
- Easier and avoids errors.

### Binding

- **Interface Definitions**
  - From app or written in an Interface Definition Language (IDL)
  - Compiles into Proxy and Stub programs
  - Client calls the Proxy (representing the server)
  - Stub calls the Server (represents the client on the server)

#### Marshaling
- Proxy marshals (sequentially lays out) calling parameters in a packet and decodes marshaled return values.
- Stub decodes marshaled calling parameters and marshals return parameters.

#### Communications Binding
- Client finds the server location via a directory service, based on server name and possibly a parameter value.
- To load balance across identical servers, randomly choose a server.

### Binding (cont'd)

- The binding process has security guarantees
  - The client must have privileges to bind to the server.
  - The client must know it's binding to an appropriate server to avoid being spoofed.
  - E.g., client and server authenticate each other during session creation, and may per-access too.

### RPC Walkthrough

- Client: Application, Proxy, RPC Runtime, Call packet
- Server: Stub, Application

### Performance

- There are basically 3 costs
  - Marshal and unmarshal
  - RPC runtime and network protocol
  - Physical wire transfer

- In a LAN, these are typically about equal.
- Typical communication times are 10-15K machine instructions.
- Can do much better in the local case by avoiding a full context switch.

### Stateful Applications

- Sometimes an application maintains state on client’s behalf, possibly across transactions.
  - E.g.,
    - Server scans a file. Each time it hits a relevant record it returns it. Next call picks up the scan where it left off.
    - Web server maintains a shopping basket or itinerary, etc.

- **Approach 1**: Client passes state to server on each call, and server returns it on each reply. Server retains no state.
  - Doesn’t work well for TP, because there’s too much state.
  - Note that transaction id context is handled this way.
Stateful Servers (cont'd)

Approach 2: server maintains state, indexed by client ID (or cookie). Client's later RPCs must go to same server.
- If the client fails, server must be notified to release client's state or deallocate based on timeout.
- For transaction RPC, encapsulate context as a (volatile) resource. Store the state at commit. Or possibly, maintain state across transaction boundaries, but reconstruct if server system failure.

E.g., COM+: Client can call a server object many times
- Client creates server object, which retains state across RPCs
- `SetComplete` (or `SetAbort`) by server app says that transaction can be committed (or aborted) and state can be deleted
- `EnableCommit` (or `DisableCommit`) by server app says transaction can (or cannot) be committed by client and don't delete server state.

Fault Tolerance

- If a client doesn't receive a reply within its timeout period
  - RPC server can send a "ping" for non-idempotent calls
  - A few multiple pings, it returns an error.
- For idempotent calls, RPC server can retry the call

Web Services

- A new round of distributed computing standards to enable interoperability on the Internet:
  - SOAP - RPC with XML as marshaling format and WSDL as interface definition
  - UDDI - directory for finding Web Service descriptions
  - WS-Transaction - 2PC
  - WS-Security, WS-Coordination, WS-Routing, ...
  - www.ws-i.org

Summary

- Scalability - 2 vs. 3 tier, sessions, stored procedures
- Web Server - gathering input, validating input, caching, authentication, constructing requests, invoking applications, load balancing
- Transaction back-end - transparency, nesting, exceptions, request integrity, savepoints
- Server processes - threads
- RPC - binding, stateful servers