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

CSE 593 Transaction Processing
Philip A. Bernstein

Outline
1. Introduction
2. Two-Tier vs. Three-Tier
3. Presentation Servers
4. Web Servers
5. Transaction Bracketing
6. Processes and Threads

2.1 Introduction
• A request is a message that describes a unit of work for the system to execute
• An application server coordinates the flow of requests between message sources (displays, applications, etc.) and application programs that run requests as transactions.
• Basic control flow:
  – Translate the display input (form/menu selection, etc.) into a standard-format request.
  – Send the request to the appropriate server based on the transaction type in the request header
  – Start the transaction
  – Invoke the transaction type’s application program
  – Commit and send the transaction’s output to the display

Application Server Architecture
• App server should make the previous control flow scale up
• Bold lines carry request messages

Application Server Components
• Web Browser
  – A smart device, with forms, menus, input validation
• Web server
  – Performs front-end work, e.g., security, data caching, ....
  – “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 Architecture (2)
• Some requests are received from other sites, rather than from a web browser
Application Server Architecture (3)
- Asynchronous requests can be queued for later processing, which may not yield an end-to-end reply (e.g. mail a bill)
- Popular with legacy TP back end systems

Application Server Architecture (4)
- Some transactions may need to enqueue other requests or send requests to other (internet or intranet) sites.

Application Server Architecture (5)
- The complete picture

Application Server Functions
- Glue and veneer for TP applications
  - glue fills in gaps in system functionality
  - covers the interface with a seamless veneer
- Provides run-time functions for applications (workflow control and transaction servers).
  - OS functions: threading and inter-process communication, often passed through from underlying OS platform.
  - Dis’d system functions: transactions, security, queueing, name service, …
  - Portal functions: Shopping cart, catalog management, personalization, …
- Provides some application development tools

Application Server Functions (cont’d)
- Provides system mgmt for the running application.
  E.g., interprets OS and DBMS metering in TP terms
  - fault monitoring and repair (fail over), server mgmt
  - security management
  - performance monitoring and tuning (load balancing)

Application Server Products
- BEA Tuxedo
- BEA Weblogic
- Bluestone Sapphire/Web
- ColdFusion
- Compaq (Tandem) Pathway
- Compaq (DEC) ACMS
- IBM CICS
- IBM IMS/DC
- IBM Websphere
- Iona iPortal App Server
- iPlanet (Sun/Netscape)
- Microsoft COM+ (formerly MS Transaction Server, or MTS)
- Oracle Application Server
- SilverStream
- WebObjects
- And many others. See serverwatch.internet.com
2.2 Two-Tier vs. Three-Tier

- Before the web, most small-to-medium scale apps were implemented in 2 tiers
- 2-tier on a LAN
  - Connect PCs directly to transaction servers
  - Server system includes transaction server app and DBMS
- Presentation server is usually a 4GL on a PC: Visual Basic, PowerBuilder, Delphi, … (RAD = rapid app development)

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) / Java Server Page (JSP))
  - Page (file) extension tells the web server to run the ASP 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 toward standard languages, such as Java
- Implement transaction server as stored procedures
- Use DBMS client-server protocol
- No application server
  - Just a 4GL plus DBMS
  - Hence, sometimes called “TP lite”

Scalability Problem of Two-Tier

- 2-tier is feasible, but does not
  - match OO application designs, which are inherently 3-tier
  - 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 (comm addr’s, authenticated user/device)
  - Standard DB APIs (ODBC and OLE DB) work this way
  - Hence, in 2-tier, N clients and M servers ⇒ N×M sessions
- Example - 10^5 presentation servers and 100 servers
  ⇒ 10^7 presentation sessions per server ⇒ 10^7 sessions overall
- Now guess why http has no sessions.

Use a Middle Tier to Reduce the Number of Sessions

- Recall 10^5 presentation servers and 100 servers
  ⇒ 10^7 presentation sessions (PS) per server
  ⇒ 10^7 sessions overall
- Now, add 100 workflow controllers (WFC)
  - Partitions the set of presentation devices (10^5 PS’s per WFC)
  - So each WFC has 10^5 PS’s and 10^7 server sessions
  - 100 WFC × (10^5 + 10^7) = 110,000 sessions
- Each transaction server has 100 WFC sessions
  - 100 TS × 100 WFC-sessions/TS = 10,000 sessions
  ⇒ 120,000 sessions overall (≈ .012 × 10^7)
Two-Tier vs. Three 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
  - Nonstandard stored procedure language, usually less expressive
    with weaker development tools and it’s another language to learn
  - Limited interoperability of cross-server calls
  - Limited interoperability of distributed transactions
- TP monitors
  - more system complexity

How the Web Changed Things

- Presentation server ⇒ Web server
- All requests have to pass through a Web server
  - Each Web server needs sessions to all DB servers
  - At least one DB session per active Web server
  - So session reduction by workflow control is still useful
- Workflow control layer is still useful for other request mgmt functions …
  - Calling Start, Commit, and Abort
  - Encapsulating business rules that transform each request into calls on basic objects

2.3 Web Servers

- Presentation independence - application is independent of the display device used
  - Today, this is via http and html
  - In the past, it was via a display controller or middle-tier minicomputer whose 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)
- 30 year evolution of presentation devices
  - Teletype, character-at-a-time terminal (async), block-mode terminal (IBM 3270), PC, web browser
  - Specialized devices - ATMs, bar code readers, gas pumps, robots, credit card authorization, cash registers, ticket printers, etc.
- Presentation tool to design and render screens
  - Forms manager - WYSIWYG form editor, compile form, execute it (with record defn binding to programming language)
  - RAD tool - ActiveX controls accessed from Visual Basic (VB), PowerBuilder, Delphi, etc.
  - Now HTML forms, and XML with XSLT

Caching

- Every process-to-process call has a cost
  - Adds to response time and consumes resources
- Use a DB cache to avoid calling workflow controller or DB system
  - Some dynamically generated pages need not be refreshed frequently
  - E.g., popular 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, department numbers
- Avoids wasting communications and server resources for obvious input errors
  - Fewer round-trips to the DBMS
  - And faster feedback to the end user
- “Cache” is part of the web page
  - List boxes, script
  - Cache size is a factor (it affects page access time)
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 OO setting
  – Request type – name of transaction type requested
  – Request-specific parameters
• In http, can be a combination of http header elements and method parameters,
• It’s helpful if each request includes a request id
  – to ask later about status,
  – to cancel a request, or
  – to allow an asynchronous reply

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
    (on the Web, use https = http over secure socket layer)
• Geographical entitlement - check that a particular device is allowed access (e.g., security trading room)
• Need system mgmt functions to create accounts, initialize passwords, bracket hours of access (simplify it using a role abstraction … more later)
• Major activity in TP application development

Application Invocation
• Request arrives as an http message.
• Needs to call a program, to perform the request

How to Call a Program
• Common Gateway Interface
  – Write a script, store it as a file in cgi-bin
  – Web server creates a process to execute the request
  – S.L.O.W
• ISAPI (Microsoft) and NSAPI (Netscape)
  – Web server calls an in-proc .dll instead of creating a process
  – More complex programming model, but much faster
  – Microsoft’s web server (IIS) caches the .dll
• Active Server Pages (and Java Server Pages)
  – Offers the performance of ISAPI with programmability of CGI

Load Balancing
• Enable scale out at all levels of the system, so you can just add more server boxes to handle more load.
• There are two related issues
  – Spreading the load evenly
  – Routing the work to the correct server
• To simplify 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
• We’ll deal with more complex scenarios later

Other Presentation Functions
• Logging
  – Record all messages sent and received by a web server
  – Useful for auditing and error analysis
• System management
  – Expose presentation and web components as meaningful concepts (e.g., named devices instead of IP addresses)
Portal Services
- Generic applications for common e-commerce tasks
- Personalization
  - user mgmt – DB app for user information
  - profile mgmt – per-user context/content/organization …
  - content mgmt – tool for defining content & format translation
  - rule engine for selecting content
- Catalog
  - API for manipulating catalog info (search, discount, report, …)
  - Pre-defined DB schema, with import/export
- Data analysis / data mining
- System mgmt UI for all the apps

2.4 Transaction Bracketing
- For the most part, Workflow Controllers and Transaction Servers are just plain old server programs
- The main differentiating features of Workflow Controller are that it
  - Brackets transactions (issues Start, Commit, and Abort)
  - Handles Aborts and other failures
  - Does not access the DBMS

Programming Languages
- Can use special purpose TP languages
  - Latest incarnation is Enterprise Java Beans (EJB) – BEA Weblogic, IBM Websphere, …
  - Older examples are Digital’s ACMSxp (Structured Txn Defn Language) and Tandem Pathway (Screen COBOL)
- Or integrate runtime library with many languages
  - IBM’s CICS, Oracle App Server, Bluestone Sapphire/Web, MS COM+, …
- Encapsulate runtime library as a container object.
- Main technical issue - TP services require runtime support.
  - It’s hard to provide nice integration with multiple languages
  - Sometimes, transaction functions are expressed in special language from which other languages can be invoked (e.g., ACMSxp).

Transaction Bracketing
- Workflow controller brackets the transaction with Start, Commit, Abort.
- Chained - All programs execute in a transaction. A program can commit/abort a transaction, after which another transaction immediately starts
  - E.g., CICS syncpoint = Commit&Start
  - Prevents programmer from accidentally issuing resource manager operations outside a transaction
- Unchained - Explicit Start operation, so some statements can execute outside a transaction
  - No advantages, unless transactions have overhead even if they don’t access resources.

Transparent Transaction Bracketing
- Transaction-hood may be a property of the application component.
- In COM+, a class is declared:
  - requires new - callee always starts a new transaction
  - required - if caller is in a transaction, then run in caller’s transaction, else start a new transaction
  - supported - if caller is in a transaction, then run in caller’s transaction, else run outside of any transaction
  - not supported - don’t run in a transaction

COM+ Transaction Bracketing (cont’d)
- Caller can create a transaction context, which supports Commit and Abort (chained model).
- SetComplete - object is finished with transaction’s work (not just this call). Commit now, if method was called outside of a transaction.
- SetAbort - abort the transaction
Enterprise Java Beans

- A Java API for Application Services
- For the most part, implements COM+ technology in Java. Came later, so there are some additions.
- EJB Transaction functions taken from COM+
  - RequiresNew, Required, NotSupported, Supports
- EJB adds
  - 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
- Some other EJB features at the end of this chapter.

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
- If (1), then be careful not to start a transaction from within a transaction
  - E.g., only start transaction in workflow control, not in a transaction server
  - (3) avoids this problem

Exception Handling

- Workflow control brackets the transaction, so it must say what to do if the transaction aborts
- An exception handler must know what state information is available
  - cause of the abort, e.g., a status variable
  - possibly program exception separate from abort reason
  - for system failures, application must save state in stable storage; note that none of the aborted txn’s state will be available
- Chained model - exception handler starts a new txn
- COM+ - component returns a failure hresult

Integrity of Request after Abort

- To permit request retries, it’s useful if get-request runs inside the request’s transaction:
  Start;
  get-request;
  . . .
  Commit;
- If the transaction aborts, then get-request is undone, so the request becomes available for the next get-request.
- In the RPC or “push model,” make the “catch-the-call” operation explicit, so it can be undone. Possibly hidden in the dispatch mechanism. Often requires a queue manager.

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.)
- Usually supported by SQL DBMSs, since it helps them support atomic SQL statements.
  Start;
  get-request;
  Savepoint("B"); . . .;
  if (error) {Restore("B"); . .; Commit;}
  . . ;
  Commit;

Savepoints (cont’d)

- Savepoints are not recoverable. If the system fails or the transaction aborts, the txn is completely undone.
- Persistent savepoints have appeared in the research literature, but aren’t supported commercially. There are formidable technical difficulties.
2.5 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 timesharing.
• 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
• Often, the Application Server implements multithreading
  – Application Server switches between threads when app calls a
    Application Server function that blocks
  – So the OS thinks there’s just one thread
  – All app calls that can block must go to the Application Server,
    or the whole process will block
  – Possible conflict between Application Server and OS scheduling
  – Simplify the implementation by using an interpretive language
    (e.g. ACMSp STDL, Tandem SCOBOL)

Multithreading (cont’d)
• Or use OS multithreading
  – No problem with blocking operations
  – Can run a process’s threads on different processors (SMP)
  – Possibly more expensive to switch threads (system calls)
• Whether at the user or OS level, multithreading has
  – fewer processes
  – reduced context switching
• Disadvantages
  – Little protection between threads
  – Server failure affects many transactions

Server Pools
• Use a set of processes (server pool or server class) to emulate multithreading
  – Better protection and fault isolation than multithreading
  – Avoids problems of user-level multithreading - blocking
    operations and conflicts between scheduling levels
• How to dispatch calls?
  – Randomly select a server
  – Server class shares a dispatch queue.
    Clients enqueue, servers dequeue.
  – Use a dispatch process (adds a context switch per call)
• Number of servers is proportional to number of active
  transactions, so not too many processes

Mapping Servers to Processes
• Presentation/Web servers - separate processes
• Workflow controllers
  – Usually multithreaded, serving many presentation servers
  – If single-threaded, then 1 per presentation server (2-tier)
• Transaction servers
  – Multithreaded servers, or
  – Server pools
• What does all this cost?
  – 1500 - 25,000 instructions per process call, vs.
    50 instructions per local procedure call …
  – but it scales, with flexible configuration and control

OO System Abstractions
• An application executing in a thread is abstracted as an
  object (e.g., COM+ object, EJB Session Bean)
• Typically, lots of active objects (threads) per class
• Such objects may have volatile state
  – Global state (e.g., a catalog) ⇒ each user’s request can be
    processed by any object
  – Per-user state (e.g., a shopping cart) ⇒ a user’s calls must all
    go to the same object (cf. TP Communications chapter)
• EJB also offers an Entity Bean abstraction, which
  represents a persistent object.
  – Persistence can be managed by the class (by implementing
    EJB-defined functions) or its container (which maps persistent
    data fields to the DBMS).
Summary

• Scalability – 2 vs. 3 tier, sessions, stored procedures
• Web Server – gathering input, validating input, caching, authentication, constructing requests, invoking applications, load balancing, portal services
• Transaction bracketing – transparency, nesting, exceptions, request integrity, savepoints
• Server processes – threads, server pools

• What’s missing? – Inter-process communications and its effect on state management