6. Application Server Issues for the Project

CSEP 545 Transaction Processing for E-Commerce

Philip A. Bernstein

Copyright ©2007 Philip A. Bernstein
Requests

• 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

Web Browser

http

Web Server

Request Controller

Transaction Server

intranet

other TP systems

Resource Manager

Resource Manager
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 request controller

• Request controller (= Workflow Controller in the project)
  – Calls Start, Commit, and Abort
  – App logic that transforms the 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)

• Resource Manager – usually a database (DB) system
Project’s Process Architecture 1

- Web Browser
- Web Server
- Request Controller
- Transaction Server
- Resource Manager
- Workflow Controller
- Resource Manager
Request Controller

- For the most part, Request Controllers and Transaction Servers are just plain old server programs
- The features that differentiate a Request Controller are that it
  - Brackets transactions (issues Start, Commit, and Abort), so that transaction server procedures can execute either as independent transactions or as steps in larger transactions
  - Reports Commits to the client (e.g., web server)
  - Handles Aborts and other failures (e.g., re-runs the transaction)
  - Does not access the DB system, so it need not be close to the DB system (i.e., Resource Manager)
Transaction Server

• The features that differentiate a Transaction Server are the inverse of the Workflow Controller, namely that it
  – Does not issue Start, Commit, and Abort (so it can be called either as an independent transaction or as a step in larger transaction)
  – Does not talk directly to the client (e.g., Web Server)
  – Can access the DB system
• In addition, it can call other transaction servers.
• Often, some transaction server code runs as stored procedures inside the DB system.
  – So combining the transaction server and resource manager in the project isn’t really an oversimplification.
Transaction Manager (TM)

- The TM is the server that supports Start, Commit and Abort.
- It implements two-phase commit (2PC).
- This is a major feature of many application servers.
  - 10 years ago, it was *the* major feature (TM + T-RPC).
  - Supports 2PC across different RMs.
  - So it’s useful to have a TM in the application server even though DB products implement 2PC themselves.
Project’s Process Architecture 2

- Web Browser
- Web Server

Request controller

- Transaction Manager
- Transaction Server
- Resource Manager

Client

Workflow Controller

Resource Manager

Start, Commit, Abort

2PC
Remote Procedure Call (RPC)

- Within a system or intranet, RPC is the most popular form of inter-process communication
- A program calls a remote procedure (in another process) the same way it would call a local procedure
  - This simplifies the message protocol. It’s always a call message followed by a return message.
  - It hides certain communications errors.
  - It automates the work of marshaling parameters into and out of the call and return messages.
- There are many implementations of the concept
  - RMI, DCOM, CORBA/IIOP, HTTP, SOAP, ODBC, ....
- In the project, all inter-process communications is via RPC.
Transactional RPC

- *Transactional RPC* is an RPC protocol that implements the necessary plumbing to cope with a caller and/or callee that are running a transaction.

- Ideally, Start returns a transaction ID that’s hidden from the caller in a *transaction context*
  - Transactional RPC passes that transaction context as a hidden parameter. It’s an easier programming model and avoids errors.
  - When a transaction first arrives at a callee C, C needs to *enlist* with the local transaction manager (TM), so the TM knows to call C during two-phase commit.
  - Also, C needs to execute the call in the context of the transaction that called it.
Transactional RPC in the Project

• You are implementing transactional RPC in the project.
  – In steps 6 and 7
  – But the transaction context parameter is explicit (not hidden).
Project’s Process Architecture (revisited)

- **Web Browser**
- **Web Server**
- **Request controller**
- **Transaction Server**
- **Resource Manager**
- **Workflow Controller**
- **Resource Manager**

- **Transaction Manager**
  - Start, Commit, Abort
  - 2PC
  - enlist 2PC

- **Client**
Partitioning Servers

- To add system capacity, add server machines.
- Sometimes, you can just relocate some server processes to different machines.
- But if an individual server process overloads one machine, then you need to partition the process.
  - Example – flights, cars, and hotel rooms are managed by one server process. Later, you partition them in separate processes.
  - This implies the WFC has to direct its RPC calls based on resource type
  - To facilitate such changes, the mapping of resource name to server name can be made table-driven.
- This scenario is developed in step (7) of the project, where multiple RMs are required.
Parameter-Based Routing

• Sometimes, it’s not enough to partition by resource type, because a resource is too popular
  – Example: flights

• The solution is to partition the popular resource based on value ranges
  – Example – flight number 1-1000 on Server A, flight number 1000-2000 on Server B, etc.
  – This implies that a request controller has to direct its calls based on parameter value (e.g. flight number)
  – To facilitate such changes, the mapping of parameter range to server name can be made table-driven.

• This is a possible project extension (not required)
Summary of Concepts

- Request Controller vs. Transaction Server
- Remote Procedure Call (RPC)
- Transactional RPC
- Transaction Manager
- Partitioning Servers
- Parameter-Based Routing
- There’s a lot more to say about Application Servers and other transactional middleware. We’ll return to the topic in a later lecture.