CSE 530A
Architecture
Washington University
Fall 2013
3-Tier Architecture

- **Presentation Layer**
  - Handles all user interaction
- **Logic Layer**
  - Encapsulates all application logic
- **Data Layer**
  - Handles persisting of data
3-Tier Architecture

• Example of *separation of concerns*

• Historically,
  – Presentation layer was a client program running on a local workstation, which
  – Connected to an application server, which
  – Connected to a database server
3-Tier Architecture

• Modified for web applications
  – Presentation layer is HTML/Javascript in the web browser
  – Middle layer is a web application
    • e.g., Java servlet, ASP .NET, Ruby on Rails
  – Data layer is a DBMS
3-Tier Architecture

• Can also refer to software architecture within an application
  – Presentation layer handles marshaling and unmarshaling of HTTP or AJAX requests and responses
  – Logic layer performs work
  – Data access layer handles interfacing with the DBMS
3-Tier Architecture

• Although in-application database engines can be used (e.g., SQLite), often a separate, standalone DBMS is used
  – Potentially running on a different physical or virtual server
PostgreSQL

- PostgreSQL runs as a standalone process (or daemon)
  - So does Oracle, MS SQL Server, IBM DB2, MySQL, etc.
- Communication is done over UNIX or TCP sockets
  - By default, PostgreSQL listens on port 5432
TCP Sockets

• A server process will **bind** to a specific port and **listen** for incoming connections
• A client process will **connect** to a server by IP address and port number
• The server process will **accept** the incoming connection
  – At this point there is a live socket connection between the client and the server
TCP Sockets

• Once connected, the server has several options
  – Handle the connection in the main thread of execution
    • Simple, but other connection attempts will block or fail
  – Use `select` to wait for activity on any socket (either the connected sockets or the listening socket)
    • Event driven, but still need to handle activity either synchronously or asynchronously
  – Spawn a thread to handle the connection. Main thread can go back to accepting connections
    • Simple thread per connection, but now have complexity of multithreading
  – Fork a new process to handle the connection
    • Eliminates dangers of multithreading, but more overhead for a process per connection
TCP Sockets

• Thread-per-connection or process-per-connection servers often use a thread or process pool to handle connections
  – Saves the overhead of always creating new threads/processes, but more complicated to implement

• A client could also keep a pool of open connections
  – Saves the overhead of always making new connections, but ties up server resources
PostgreSQL

• PostgreSQL uses the process-per-connection, fork-on-demand approach
  – Therefore, it is often better for a client to reuse an open connection than to close and open a new connection
  • Though this ties up a connection on the server
APIs

• Standard APIs are used by applications to communicate with DBMSs
  – ODBC: Open Database Connectivity
  – JDBC: Java Database Connectivity

• Use of standard APIs attempts to achieve DBMS independence
  – Somewhat foiled by the fact that different DBMSs use different variants of SQL
Administration

• Interaction with DBMSs can be done through administrative clients
  – From the point of view of the DBMS, just another client application
• PostgreSQL has psql
psql

• psql is a terminal interface for accessing PostgreSQL databases
  – Connects to PostgreSQL daemon
  – SQL commands can be entered and text output will be sent back
  – Also has non-SQL commands for administration
    • Describing tables, configuring connect parameters, etc.