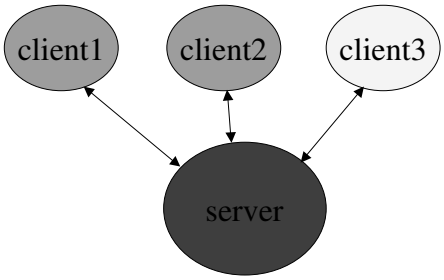
Systems Architecture

Client-Server Systems

Client/Server

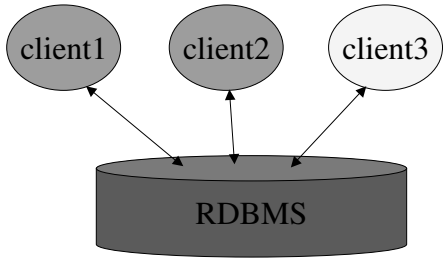
- In general, any application where multiple clients connect to a single server.



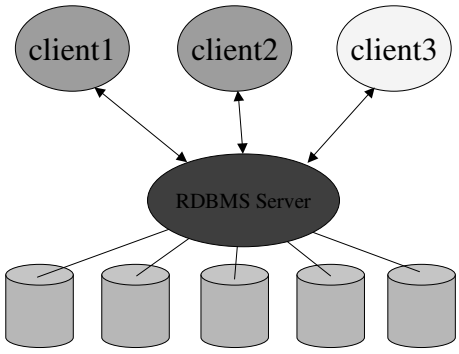
- one client program (most typical)
  or
- multiple client programs

Relational Databases

- Most common client/server program is where the server is a relational database server.
  - warning: some use the term client/server to refer to this usage exclusively (we won't).



Relation Database Implementation



disks

IPC

- "Inter-Process Communications"
  - How processes will communicate and synchronize with one-another.
  - communications mechanisms:
    - shared memory
      - very fast
      - can't use over a network
        - » well, you can
    - message passing
      - can use over a network
      - slower
        - » well, not always
  - will consider only message passing (most important)

IPC Protocols

- Basic message-passing mechanisms provide for a byte-stream only.
- Must implement various protocols on top of this
  - sockets
  - RPC (remote procedure call)
  - DO (distributed objects)

Sockets code example

```
public class Server {
    public static void main(String[] args) throws Exception {
        ServerSocket server = new ServerSocket(1234);
        Socket client = server.accept();
        BufferedReader fromClient = new BufferedReader(
                 new InputStreamReader(client.getInputStream()));
        System.out.println(fromClient.readLine());
    }
}


public class Client {
    public static void main(String[] args) throws Exception {
        Socket server = new Socket("penny", 1234);
        DataOutputStream toServer = new DataOutputStream(
                 server.getOutputStream());
        toServer.writeBytes("hello server");
         server.close();
    }
}
```


Performance

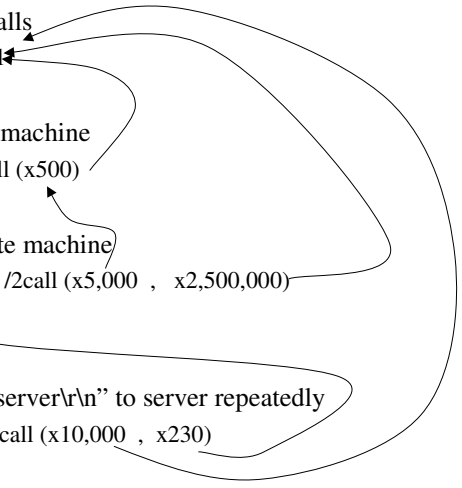
- Latency
  - The time to go back and forth
- Bandwidth
  - The amount of data that can be sent
- Analogy from ocean lines
  - Bandwidth of QE2 is high (can carry a lot)
  - Latency is bad (takes a long time for a round trip).

Test System

- Windows 2000 Java Server
  - Network
    - 100 Mbit/s ethernet
  - CPU
    - dual 1GHz processors
  - Memory
    - 1 GByte
- Windows 98 Java Client
  - Network
    - 100 Mbit/s ethernet
  - CPU
    - 366 MHz
  - Memory
    - 96 MByte

Java/Windows Performance Measures

- Latency: Sending “hello server\r\n” back and forth
  - Local method calls
    - .13 usec/2call
  - Socket on local machine
    - 70 usec / 2call (x500)
  - Socket on remote machine
    - 320,000 usec /2call (x5,000 , x2,500,000)
- Bandwidth
  - Sending “hello server\r\n” to server repeatedly
    - 1400 usec / 2call (x10,000 , x230)



Performance

	In Process	Network
Latency	1	2,500,000
Bandwidth	1	10,000

C/Windows Performance Measures

- Latency: Sending “hello server\r\n” back and forth
  - Local method calls
    - .01 usec/2call (10x Java)
  - Socket on local machine
    - 12 usec / 2call (6x Java)
  - Socket on remote machine
    - 840 usec /2call (380x Java)

Performance

	In Process	Network
Latency	1	84,000

Performance Implications

- Do as few calls as possible over the net
- Prefer asynchronous approaches
  - problem: success/failure indications
  - send lots of stuff, then synchronize
- Use bigger transactions
- Prefer one call with lots of data to many calls with the same amount of data
  - but not by much
- Send as little data as possible

Relational Databases

- Most common type of client/server software is where the server is an RDBMS server:
  - Oracle
  - SQLserver
  - Sybase
  - Informix

Relational Databases

- Stores data into tables

order		
orderid	orderdate	custid
239	Nov.13	2349
267	Nov.14	3903

customer		
custid	custname	credit
2394	Fred	ok
3903	Susan	ok
3453	Mandy	bad

orderitems		
orderid	itemid	quantity
239	12	1
239	28	4
267	42	10

items		
itemid	itemname	onhand
12	bread	2142
28	sugar	345
42	beer	2134

Database Access

- Access using SQL (Standard Query Language)
  - select itemname,quantity
    - from
      - orderitems,items
    - where
      - orderid = 239
        - » and
      - orderitems.itemid = items.itemid

"stored procedure" if this is parameterized and the whole thing is named

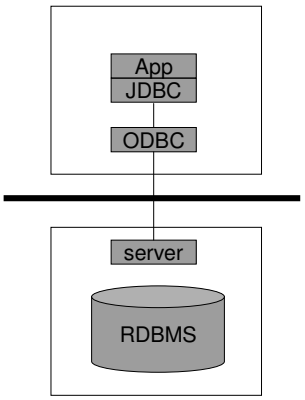
query result	
itemname	quantity
bread	2142
sugar	345

Programmatic Database Access

- Can access database by
  - typing commands at an sql command prompt
  - by running a GUI tool
  - programmatically
    - ODBC
      - Open Database Connectivity – Microsoft standard API
      - ANSI/ISO CLI is ODBC level1 compliant (Call Level Interface)
        - » (see also DAO, OLE DB and ADO)
    - JDBC
      - very similar to ODBC
    - Various embedded SQL hacks

JDBC

- All sorts of possible configurations of client-side & server-side drivers



Database Access from Java

```
import java.sql.*;
public class Main {
   private static final query =
       "select itemname,quantity " +
       "from orderitems,items " +
       "where orderid=1 and orderitems.itemid=items.itemid";

    public static void main(String[] args) throws Exception {
         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
         Connection c = DriverManager.getConnection("jdbc:odbc:grocery");
         Statement s = c.createStatement();
         if( s.execute(query) ) {
             ResultSet r = s.getResultSet();
             printResults(r);
         }
    }

     private static void printResults(ResultSet r) throws Exception {
        final int nC = printHeadings(r);
        printRows(nC, r);
    }
    …
}
```

Database Access from Java

```
private static int printHeadings(ResultSet r)
        throws Exception {

    ResultSetMetaData m = r.getMetaData();
    final int nC = m.getColumnCount();
    for(int c = 1; c <= nC; c++) {
        System.out.print(m.getColumnName(c));
        System.out.print("\t");
    }
    System.out.println();
    return nC;
}
```


Database Access from Java

```
private static void printRows(int nC, ResultSet r)
        throws Exception {
    while( r.next() ) {
        for(int c = 1; c <= nC; c++) {
            System.out.print(r.getString(c));
            System.out.print("\t");
        }
        System.out.println();
    }
}
```

Without ODBC

```
Class.forName(
    "org.gjt.mm.mysql.Driver"
    );


Connection c = DriverManager.getConnection(
    "jdbc:mysql://penny.dhcp.cs.toronto.edu/grocery"
    );
```


Performance

- localhost
  - JDBC:ODBC
    - 850 us/query
  - JDBC:MYSQL
    - 500 us/query
- over network
  - JDBC:ODBC
    - 3,800 us/query
  - JDBC:MYSQL
    - 1,600 us/query

- local Java method call
  - 0.13 us/query
- C socket over network
  - 840 us/query