Web Server Architecture

Adapted from Steve French, digg.com

Stack

Server OS

Linux,MacOS X,UNIX,Windows

Web Server

apache,iis,nginx,lighttpd,tornado

Programming Language

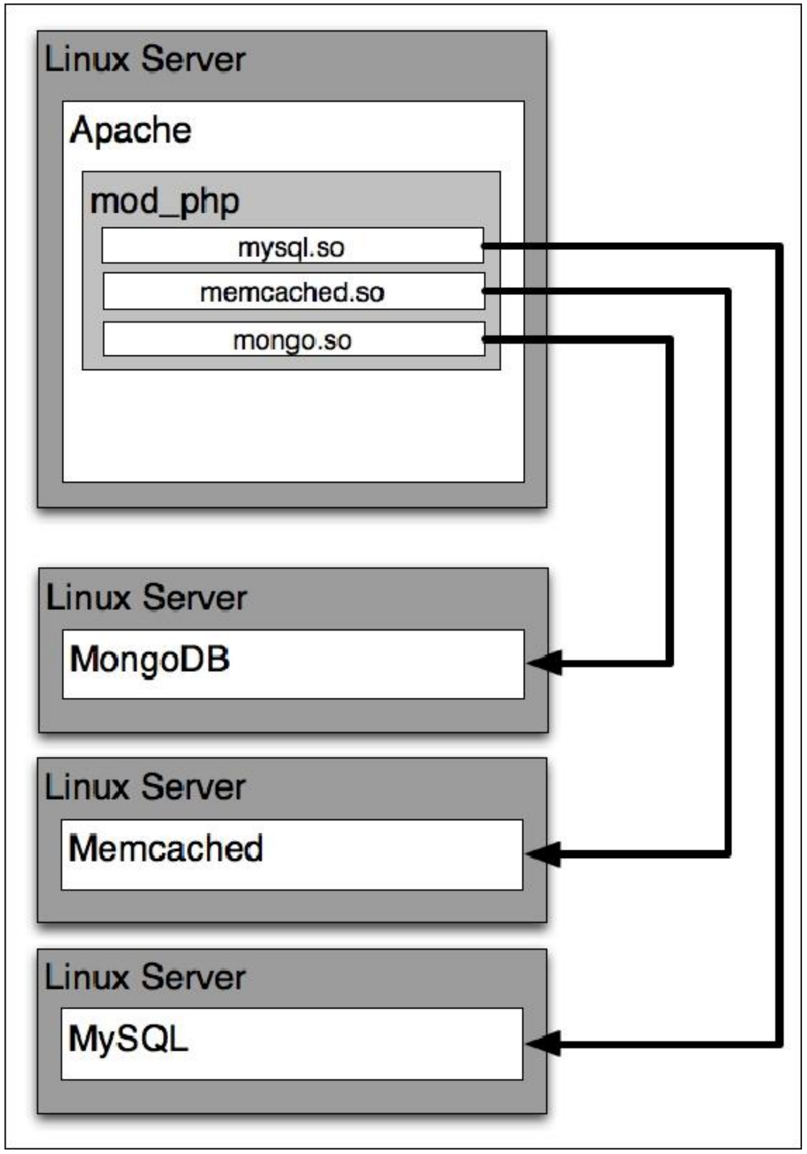
php,python,ruby,java,asp,jsp

Datastore

mysql,postgresql,oracle,db2,mssql
cassandra,mongodb,couchdb

What is Digg.com

Linux +Apache
PHP,Python,Java
MySQL,Cassandra



Reduce Load

Content Delivery Networks (CDN)

You can push static content (images/css/javascript/video) to external delivery networks which take the request load off of your servers

Add MoreWebservers

Technically this is true,but isn’t generally going to be the best option.

Tune ExistingWebservers

If your server has enough resources you can experiment with increasing the maxclients that your webserver will allow.

Webserver Load

- Optimize Code and Code Use
  - Cache Code
  - Cache Data Outputs
  - Profile Code
- Scale the SystemArchitecture
  - Scale Up
  - Scale Out
    - Many servers behind load balancer(s).
    - Some care is needed to make sure that your code does not require machine affinity.

DataAccess Load

Cache Data

- Memcache - http://www.danga.com/memcached/
  - Test:
    - Fetch 100000 users from mysql and memcached
- Caching Strategy
  - Cache OnAccess
  - Optimistic Cache Priming

Optimize Queries

- Explain Queries
- Add Indices
- Sometimes many“simple” queries are better than a single complex one.

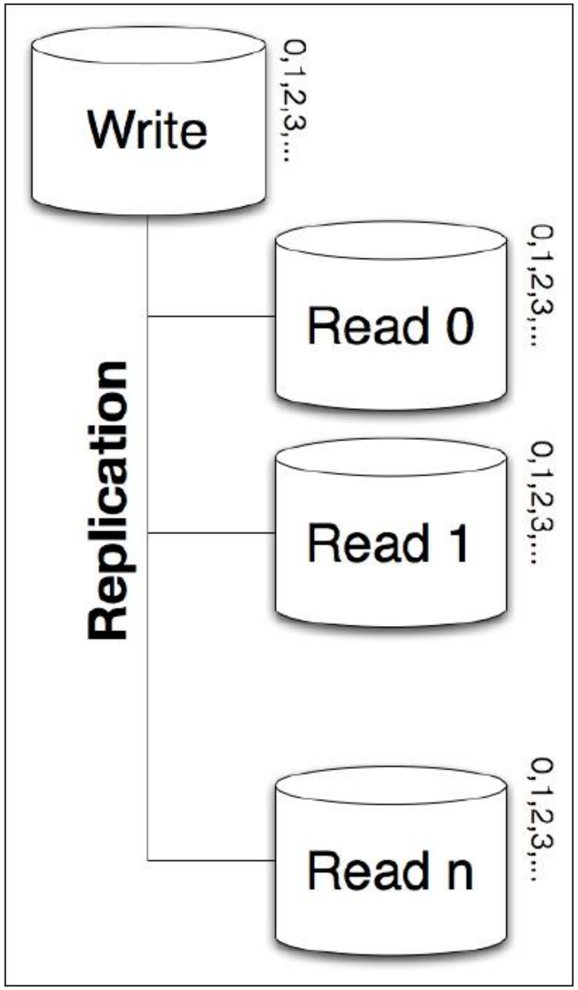
```
select * from users where userid in (1000,3002,53,100000,999,....);

    VS

for($i=1,$i <= 500;$i++)
    select * from users where userid = $i
```

DataAccess Load

Scale the SystemArchitecture - ReadTraffic

- Scaling Out Read Slaves
- Vertical Partitioning
- Horizontal Partitioning
- Data Denormalization

Scaling Out Read Slaves

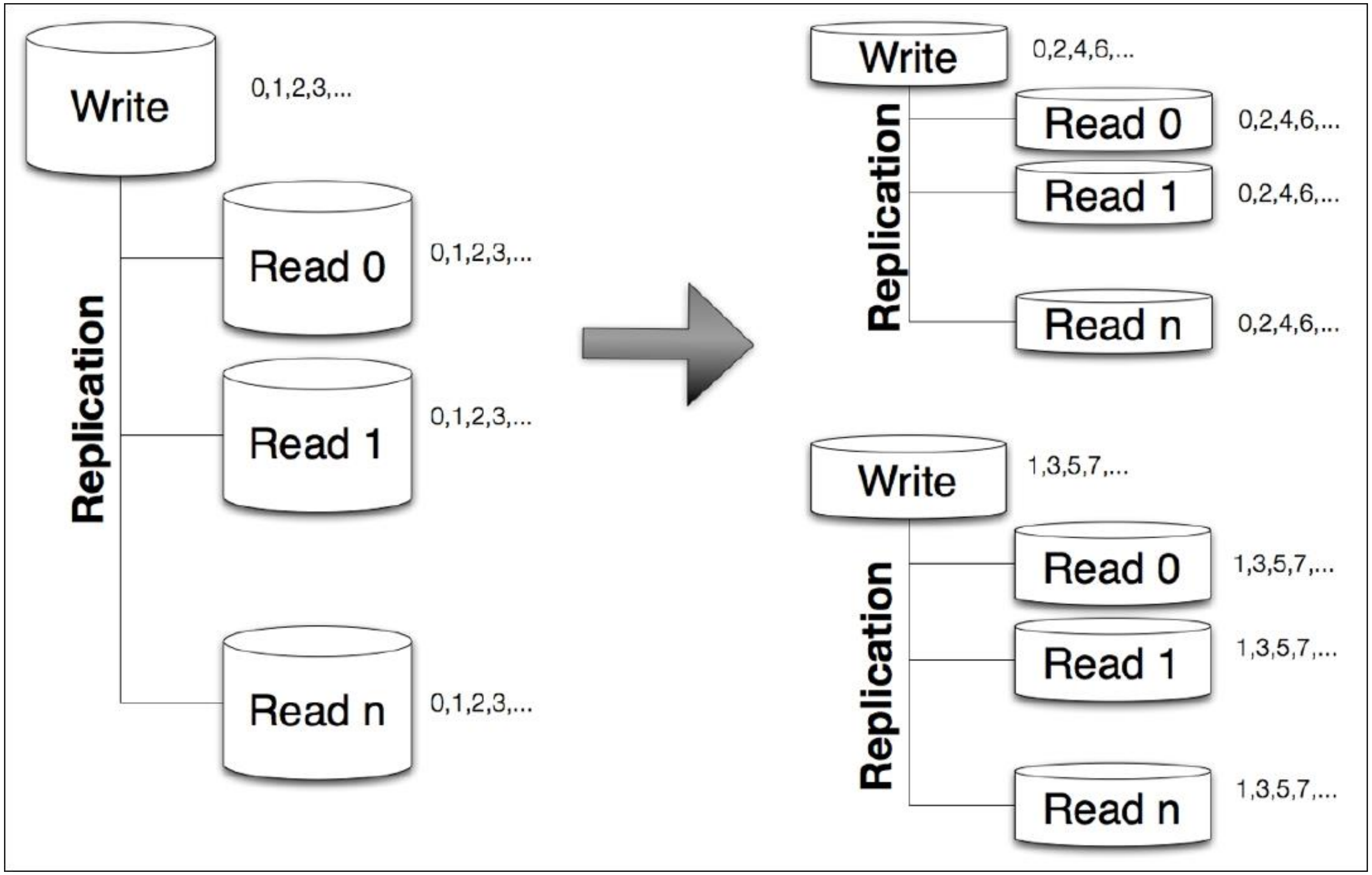


Vertical Partitions

```
create table users(
    id bigint not null auto_increment,
    username char(20),
    password char(50),
    secretquestion char(100),
    secretanswer char(100),
    favoritecolor char(25),
    website text,
    active enum('Y','N')
);
```

```
create table users(
    id bigint not null autoincrement,
    username char(20),
    password char(50),
    active enum('Y','N')
);

create table userdetails(
    userid bigint not null,
    secretquestion char(100),
    secretanswer char(100),
    favoritecolor char(25),
    website text
);
```

Horizontal Partitions (Sharding)



Data Denormalization

Benefits

- Data is replicated to tables that need it.
- Less JOIN operations

Drawbacks

- Updating values takes additional operations,so this technique is best used on rarely-changing data.

```
create table users(
    id bigint not null autoincrement,
    username varchar(20),
    password varchar(50),
    secretquestion varchar(100),
    secretanswer varchar(100),
    active enum(‘Y’,’N’)
);
create table stories(
    id bigint not null autoincrement,
    title varchar(100),
    description varchar(255),
    username varchar(20)
);
create table comments(
    userid bigint not null,
    comment_body varchar(255),
    username varchar(20)
);
```

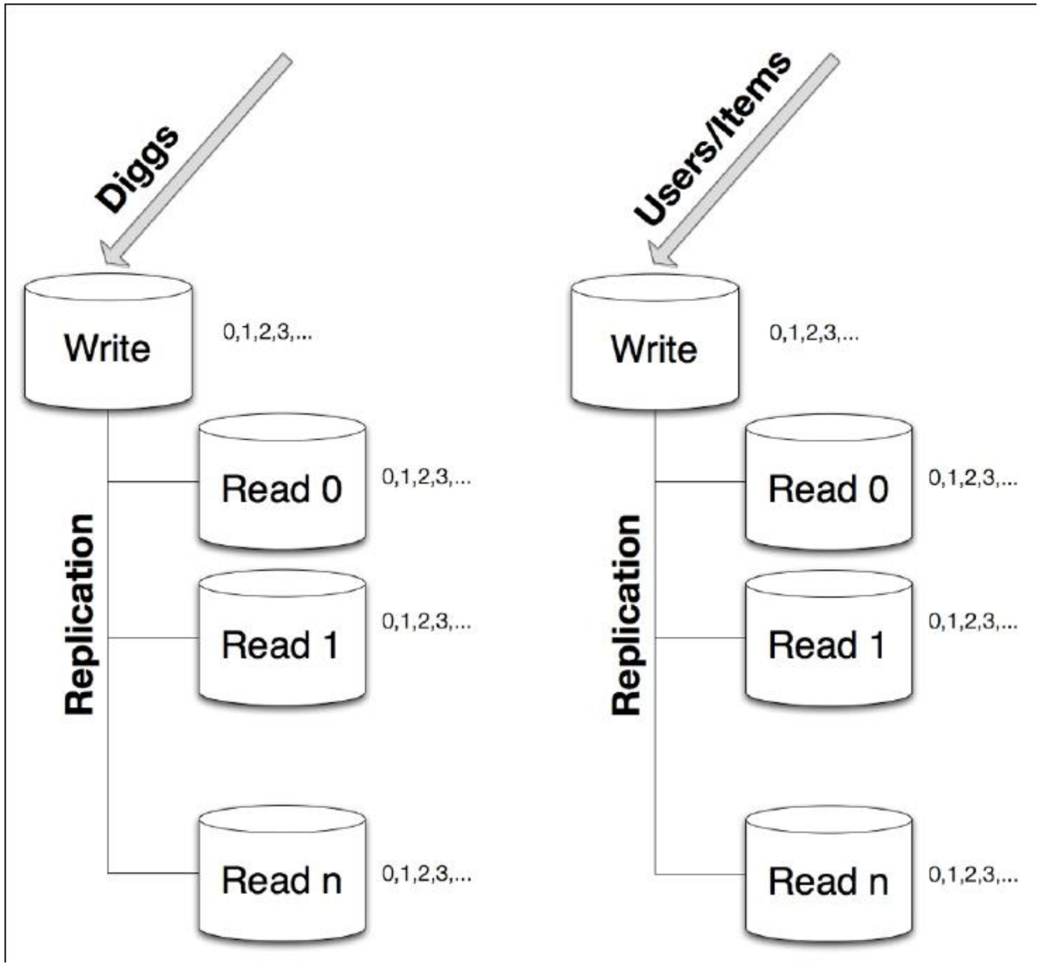
Scale the SystemArchitecture -WriteTraffic

Query-Specific Server Pools

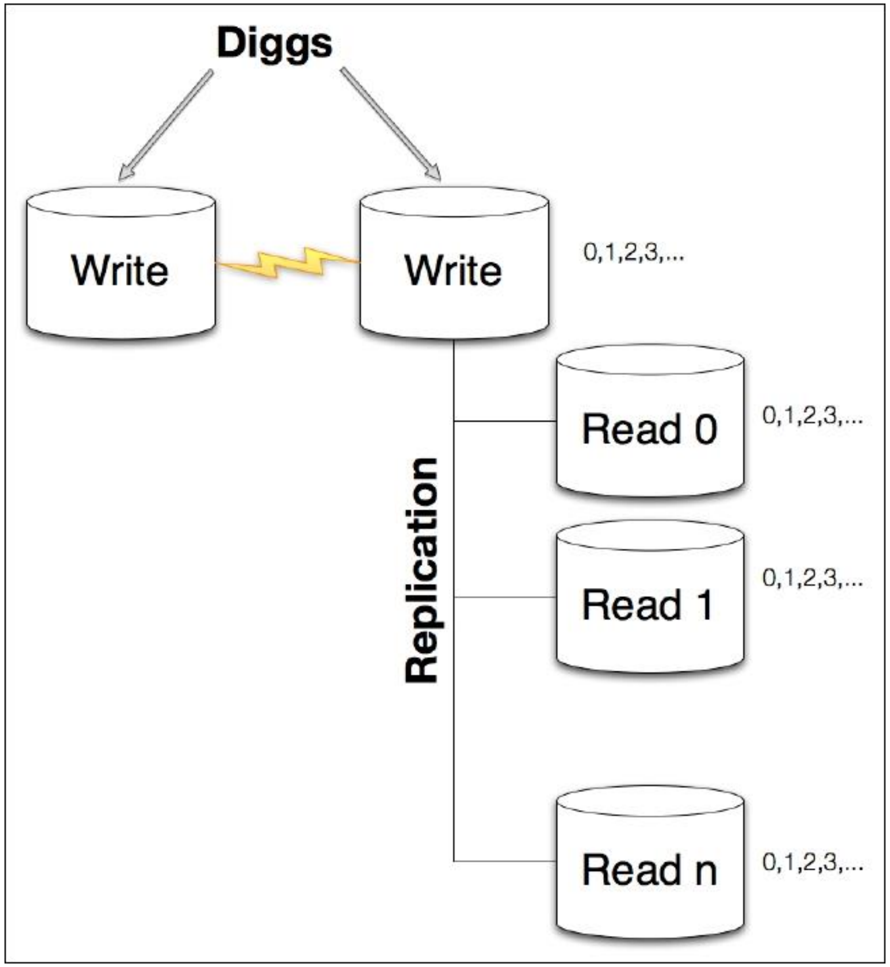
Scale OutWritable Servers

Horizontal Partitioning

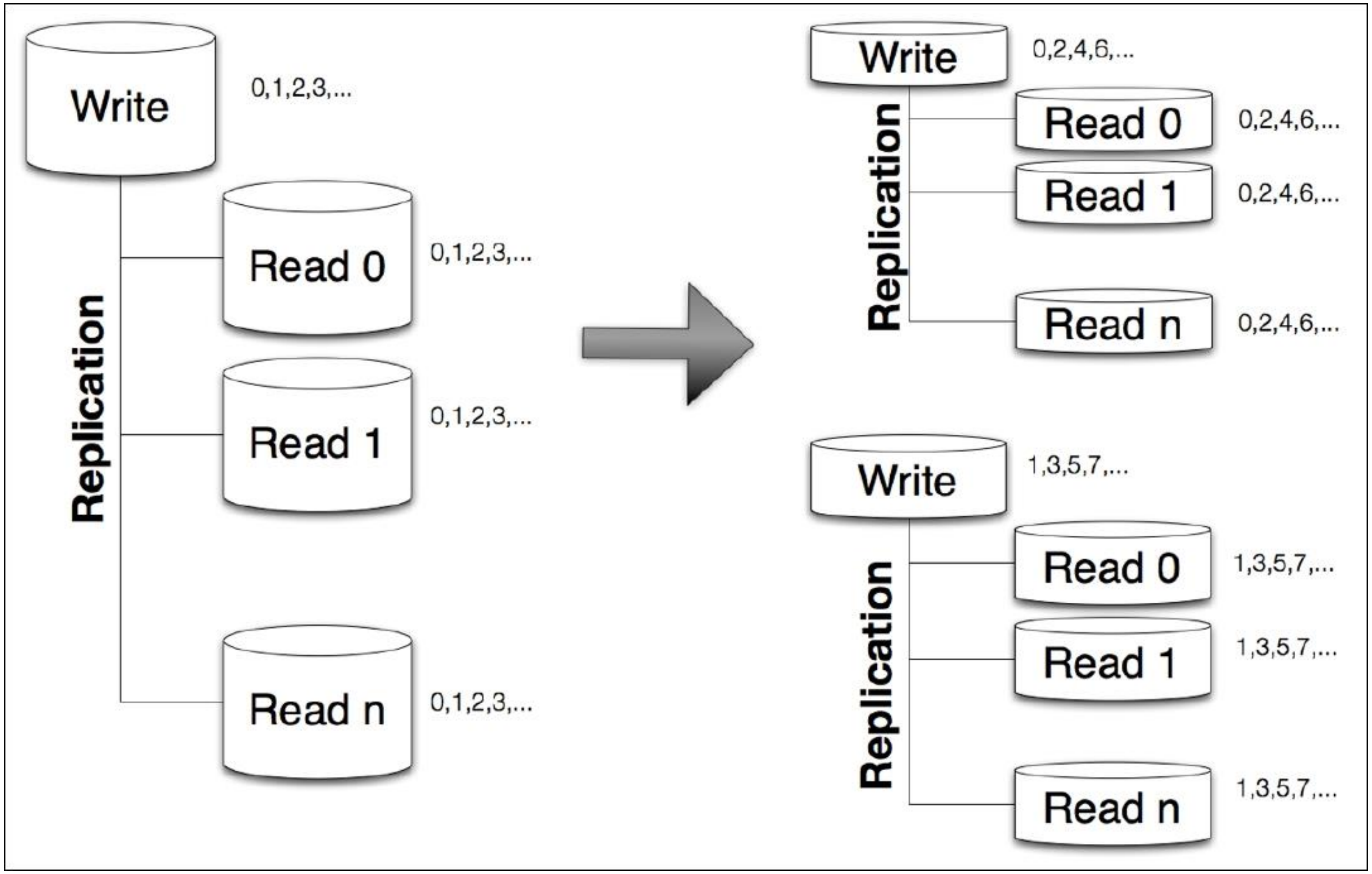
Query-Specific Server Pools



Scale OutWritable Servers



Horizontal Partitions (Sharding)



Key/Value Stores

Benefits:

Faster
Suited to horizontal scaling
Redundancy
Read/WriteTradeoffs

Drawbacks:

Ad-hoc queries are almost impossible.

Cassandra Data Model - Example

```
AddressBook = { // ColumnFamily of type Super
  webservice: {
    // all the address book entries (supercolumns) for webservice
    John: {street: "Howard street", zip: "94404", city: "San Francisco",
           state: "CA"},
    Kim:   {street: "X street", zip: "87876", city: "Sticks", state: "VA"},
    Tod:   {street: "Jerry street", zip: "54556", city: "Cartoon",
           state: "CO"},
    Bob:   {street: "Q Blvd", zip: "24252", city: "Nowhere", state: "MN"},

  },
  jms: {
    // all the address book entries for jms
    Joey:      {street: "A ave", zip: "55485", city: "Elko", state: "NV"},
    William: {street: "Armpit Dr", zip: "93301", city: "Bakersfield",
              state: "CA"},
    Bob:         {street: "Q Blvd", zip: "24252", city: "Nowhere",
              state: "MN"},
  },
}
```

Gearman

```
$users = fetch_all_users();
foreach($users as $user) {
    //do an expensive process
}

$items = fetch_all_items();
foreach($items as $item) {
    //do an expensive process
}
```

```
gearman_exec(fetch_all_users,$userkey);
gearman_exec(fetch_all_items,$itemskey);

while(!gearman_fetch($userskey))
    sleep(1);

foreach($users as $user) {
    //do an expensive process
}

while(!$items = gearman_fetch($itemskey))
    sleep(1);

foreach($items as $item) {
    //do an expensive process
}
```

Asynchronous Job Queues

Gearman - http://www.danga.com/gearman/
RabbitMQ - http://www.rabbitmq.com/
JMS - http://java.sun.com/products/jms/

Offline Processing

Cron -“man cron”
Hadoop - http://hadoop.apache.org/

Databases Security – General Strategi es

- Principle of Least Privilege!
- Stay up-to-date on patches
- Remove/disable unneeded default accounts
- Firewalling/Access Control
- Running Database processes under dedicated non-privilege d account.
- Password Security
- Disable unneeded components

Principle of Least Privilege

- If X service doesn't need access to all tables in Y databa se… then don't give it access to all tables.
  - Example: A web application that reads a list of people from a database and lists them on a website. The database also contains sensitive infor mation about those people.
- Do not give accounts privileges that aren't needed
  - Unneeded privileges to accounts allow more opportunity for privilege escalation attacks.

Stored Procedures, Triggers

- Stored Procedures and Triggers can lead to privilege escala
tion and compromise. Be sure to be thinking about securit
y implications when allowing the creation of, and creating
these.

SQL Injection

- is a technique that exploits a security vulnerability occurring in the database layer of an application. The vulnerability is pre sent when user input is either incorrectly filtered for string li teral escape characters embedded in SQL statements or user input is not strongly typed and thereby unexpectedly execute d.

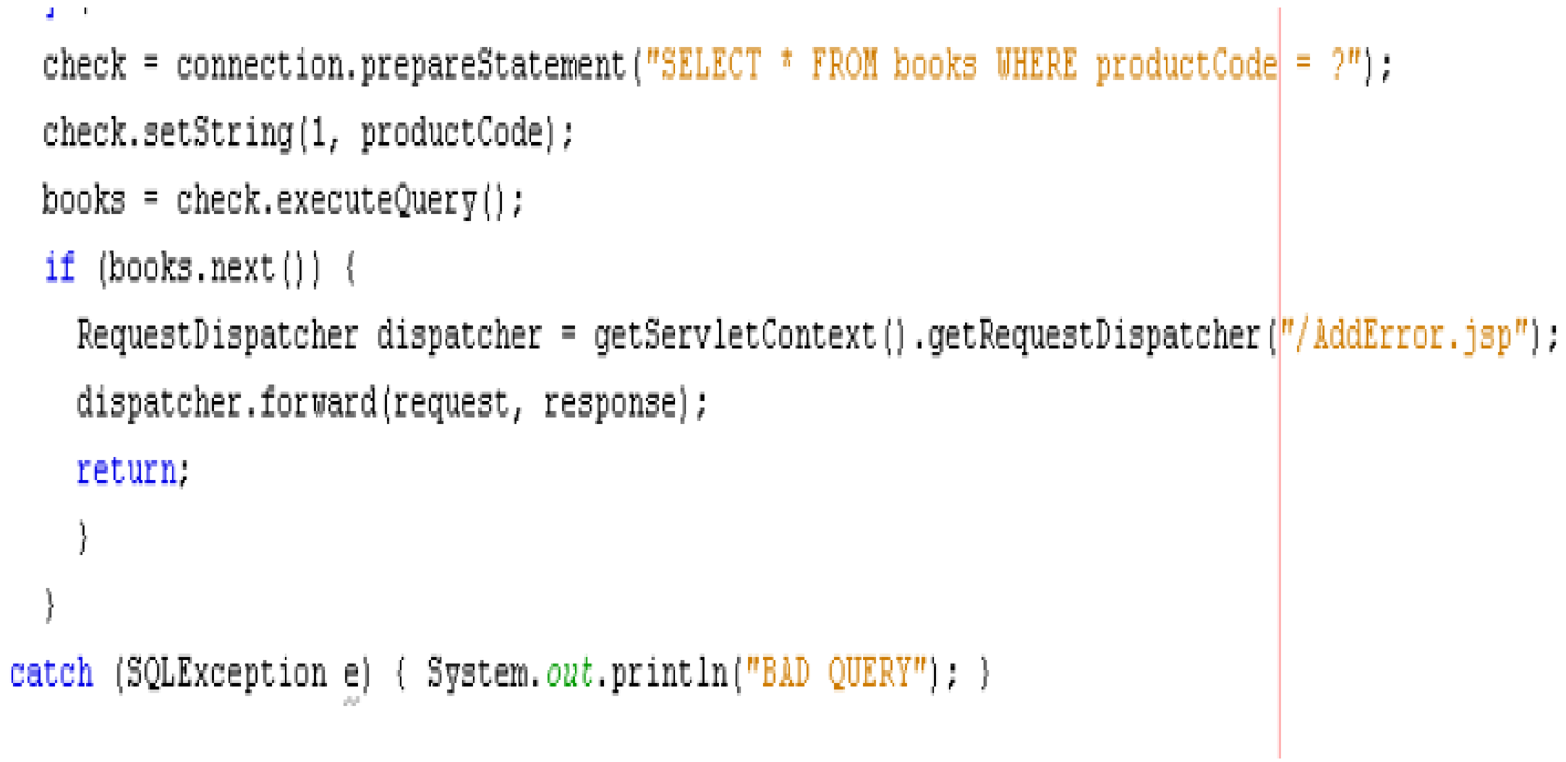
SQL Injection

- Username = ' or 1=1 --
  - The original statement looked like:
    'select * from users where username = '" + username + "' and pa
    ssword = '" + password + "'‘
    The result =
    select * from users where username = '' or 1=1 --' and password
    = ''

Prepared Statement

- Executing a statement takes time for database server
  - Parses SQL statement and looks for syntax errors
  - Determines optimal way to execute statement
    - Particularly for statements involving loading multiple tables
- Most database statements are similar in form
  - Example: Adding books to database
    - Thousands of statements executed
    - All statements of form:
    - **"SELECT * FROM books WHERE productCode = ____“**

Prepared Statement



```
  check = connection.prepareStatement("SELECT * FROM books WHERE productCode = ?");
  check.setString(1, productCode);
  books = check.executeQuery();
  if (books.next()) {
    RequestDispatcher dispatcher = getServletContext().getRequestDispatcher("/AddError.jsp");
    dispatcher.forward(request, response);
    return;
    }
  }
catch (SQLException e) { System.out.println("BAD QUERY"); }
```

Stored Procedure

- A stored procedure is a precompiled executable object that c ontains one or more SQL statements.
  - stored procedures are precompiled objects they execute faster a t the database server.
  - For the consecutive run it will run from the compiled stage and hence boost performance.

DB Encryption

- DB Encryption can be divided into Data-in-transit and Data-a t-rest
- Encryption is useful as a last layer of defense (defense in dept h). Should never be used as an alternative solution
- Encryption should be used only when needed
- Key Management is Key

Auditing

- Oracle-supplied auditing using AUDIT command. Results go to AUD$
- Trigger-based DML auditing
- Either way, DBA must monitor auditing table.

Example of Audit command

- Must have audit system privileges
- Only tracks in subsequent user sessions
- Creates records in table AUD$ owned by sys
  - You don't query this table, you query
    - Views such as DBA_AUDIT_TRAIL
- SQL> AUDIT Delete any table;
- SQL> NOAUDIT delete any table;

When to audit

- When should we audit Oracle users ?
  - Basic set of auditing measures all the time
  - Capture user access, use of system privileges, changes to the db schema (DDL)

    If company handles sensitive data (financial market, military, etc .) OR

    If there are suspicious activities concerning the DB or a user, sp ecific actions should be done.

SQL Tuning

- **Avoid the LIKE predicate** = Always replace a "like" with an equality, when appropriate.
- **Never mix data types** - If a WHERE clause column predic ate is numeric, do not to use quotes. For char index columns, always use quotes. There are mixed data type predicates:
  where cust_nbr = "123"
  where substr(ssn,7,4) = 1234
- **Use those aliases** - Always use table aliases when referencin g columns
- **Use minus instead of EXISTS subqueries** - Some say th at using the minus operator instead of NOT IN and NOT Exi sts will result in a faster execution plan.

- The sql query becomes faster if you use the actual columns names in SELECT statement instead of than '*'.
- **Example:** Write the query as
- SELECT id, first_name, last_name, age, subject FROM student_details;
- Instead of: SELECT * FROM student_details;
- Sometimes you may have more than one subqueries in your main query. Try to minimize the number of subquery block i n your query.
- **Use SQL analytic functions** - The Oracle analytic functio ns can do multiple aggregations (e.g. rollup by cube) with a si ngle pass through the tables, making them very fast for repor ting SQL

- Use operator EXISTS, IN and table joins appropriately in you r query.
  a) Usually IN has the slowest performance.
  b) IN is efficient when most of the filter criteria is in the sub -query.
  c) EXISTS is efficient when most of the filter criteria is in th e main query.
  Example: Write the query as
- Select * from product p
  where EXISTS (select * from order_items o where o.product _id = p.product_id)
- Instead of: Select * from product p where product_id IN (select product_id from order_items)

- Be careful while using conditions in WHERE clause.
  For Example: Write the query as
- SELECT id, first_name, age FROM student_details WHERE age > 10;
- Instead of: SELECT id, first_name, age FROM student_detai ls WHERE age != 10;