Written Exam 1

Fall ‘19

This exam has six (6) questions worth a total of 100 points. You have fifty (50) minutes. This exam is preprocessed by a computer when grading, so please write darkly and write your answers inside the designated spaces.

Resources. This is an open-book exam (notes, web, etc. permitted). You may not communicate with anyone except the course staff during this exam.

Discussing this exam. Due to travel for extracurriculars and sports, some of your peers will take this exam at a later date. Do not discuss its contents with anyone until the exam has been returned to you.

This exam. Do not remove this exam from this room. In the space provided, write your name and NetID. Finally, write and sign the Honor Code pledge. You may fill in this information now.

Name: ________________________________

NetID: ________________________________

“I pledge my honor that I will not violate the Honor Code during this examination.”

______________________________
Signature
Question 1 - Short Answer.  
points

Provide a one sentence answer to each question below:

What is a content-based name?

A content-based name is a name that is based on the content of the object. For example, a git commit is named with a hash of the commit’s content.

What is the difference between global and local names?

Global names are valid and unique in the global scope of a system, whereas local names are only valid and unique in the local scope (e.g., within a directory or a process’s address space) and can have duplicates in the global scope.

A social web application often accesses data from a user’s friends after accessing the user’s data. Does this access pattern exhibit locality? If so, which kind?

This access pattern exhibits spatial locality.

What is the purpose of the reference layer in Git?

Reference layer in Git provides global, human readable names like “master”, “HEAD”, “origin” etc. for objects.

What is one advantage of a write-back cache over a write-through cache?

Write-back cache can reduce the number of write operations to memory.
Question 2 - System Architecture.  

You are designing a digital library system with the following features:
- The server is available on the internet
- Clients can download digital books, share reviews, etc.
- The catalog will include over 10M books
- Performance should be tuned for the most popular titles

Your architecture leverages various mechanisms and APIs we have covered during this semester. Complete the high-level system architecture diagram, where rectangles represent various modules and layers in the system. Label each rectangle with the most appropriate mechanism/API from the list below. Each label may only be used once. One rectangle is already labeled for you.

Labels:

1. OPERATING SYSTEM  
2. CLIENT  
3. DNS & HTTP  
4. TRANSPORT  
5. NAME SERVER  
6. PHYSICAL  
7. NETWORK  
8. http.NewRequest  
10. http_router.ServeHTTP  
11. http-router.AddRoute  
12. http.ListenAndServe  
13. DATABASE  
14. SERVER  
15. LINK  
16. CACHE
Question 3 - DNS and HTTP.  

Suppose an Internet client (the node labeled “Client”) requests a web page from a popular new site: 
\texttt{http.Get("noneofyour.biz").} The site is hosted on an Internet server located at IP address 
\texttt{44.206.251.100}.

Which DNS and HTTP traffic occurs to service this request?

In the diagram below, nodes (circles) represent computers on the Internet and edges are potential 
network communications between the nodes. Associate the edges (labeled A-K) with their order of 
occurrence.

<table>
<thead>
<tr>
<th>Order</th>
<th>Edge Label</th>
<th>Order</th>
<th>Edge Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>D</td>
<td>6</td>
<td>B</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>7</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>8</td>
<td>E</td>
</tr>
<tr>
<td>4</td>
<td>H</td>
<td>9</td>
<td>G</td>
</tr>
<tr>
<td>5</td>
<td>I</td>
<td>10</td>
<td>J</td>
</tr>
</tbody>
</table>
# Question 4 - LRU and FIFO Caching.

Assume an application with ten (10) unique objects of the same size, numbered 1 - 10, and a dedicated cache (i.e., no other applications use the cache). Assume that the cache has room for five (5) objects at any given time.

What is the number of cache hits and cache misses, and the hit rate for the following trace under FIFO and LRU eviction policies?

<table>
<thead>
<tr>
<th>Eviction Policy</th>
<th># Cache Hits</th>
<th># Cache Misses</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIFO</td>
<td>4</td>
<td>14</td>
<td>4/18 == 2/9 == 22%</td>
</tr>
<tr>
<td>LRU</td>
<td>8</td>
<td>10</td>
<td>8/18 == 4/9 == 44%</td>
</tr>
</tbody>
</table>
Question 5 - Naming Analysis.  

In lecture we discussed how UNIX files are “owned” by a user, and how each file has permission bits that control what kinds of actions the owner can perform on the file (read, write and execute).

In addition to user-level permissions, UNIX groups allow a system administrator to define access policies on files for a group of users. In addition to having a user “owner,” each file also has an owning “group,” with the same set of permission bits.

If a user belongs to the group that owns a file, they may perform actions on the file as specified by the group permission bits (read, write and execute).

Groups are defined in the file /etc/groups. Each line is a colon-separated row containing:
  ● the group name
  ● a group password (you can ignore the password field, it is never used)
  ● group id (a unique integer)
  ● a comma-separated list of user belonging to the group

For example:

```
cos316:x:20819:ak18,aalevy,xiaoqic,araina,mh43,hl7
fac:   x:33012:ak18,aalevy
grad:  x:14512:xiaoqic,araina,mh43,hl7,z71
```

When a user tries to access a file, the kernel permits the access if the user belongs to the group that owns the file and if the file permissions for the group permit the requested access.

For example, suppose a file named roster.txt is owned by user ak18 and is in the fac group. The user permissions are set to read/write and the group permissions are set to read. In this case, user ak18 can both read and write the file, while user aalevy can only read the file.

Analyze UNIX groups using the naming properties we discussed in class:

  ● Names
  ● Values
  ● Allocation
  ● Translation

Provide your answer on the following page.
Outline:

There is more than one way to answer this question. The most important part is to focus on how UNIX models and stores groups.

Names in the scheme are group names: alphanumerical strings that have the same characteristics as UNIX user names, but are a different namespaces (e.g. there can be both a user and group named “cos316”). Group IDs—an integer—is a good answer here, as is group name and group id, i.e. names are a pair of (group name, group id).

Values are the list of users that belong to a particular group. Alternatively, the remaining fields in an /etc/groups row not including the name would be good too: e.g. the password, group id and user list.

Allocation of both group names and the list of users are human-assigned—an administrator or user chooses a unique group name, and decides which users to include in the group (users don't have to be unique among groups). If we define group ids as part of the name or value, we must also note that allocation is done by finding a unique integer—e.g. by using a monotonically increasing integer, or by explicitly scanning the rows in /etc/groups for all existing group IDs and choosing the lowest number not already included in the list.

Translation of group names to a list of users is done by scanning through each line in /etc/groups, splitting the line by it’s separator (“:”) and comparing the first column to the input group name. If it matches, we return the last column, which contains a comma-separated list of users.
Question 6 - Cache Analysis.  

Recall that relational, or SQL, databases structure data into structured tables. Each table has a fixed set of columns and rows containing values for each column. These databases are called relational because tables often have relationships between them. For example, two tables containing movies and actors might be related through a third table that joins actors that appeared in movies.

Web applications often use relational database to store their data because these databases are so flexible and expressive, but this flexibility can come at the cost of performance. In particular, SQL queries can be slow as compared to web application requirements. As a result many web applications cache data from a relational database to speed up in the common case.

Design a cache for web applications that query a relational database. You may make assumptions about the web application or the structure of the database, and impose constraints that are useful, but you must state these clearly.

You answer should address at least the following questions:

- What kinds of keys and values would your cache store?
- Is your cache look-aside or look-through? Write-through or write-back? write-allocate or write-no-allocate? Explain why you made each choice.

Provide your answer using the space below and the following page.

Outline:

There are many good answers to this question, and no single “correct” answer. However, good answers have to be internally consistent and well justified.

In practice there tend to be two kinds of caches used in these kinds of applications.

First, is the database's own cache.

The database cache uses SQL queries (or generated sub-queries if queries are very complex) as keys, and their results as values.

It is a look-through cache, which makes sense for the same reasons it does in a CPU cache: it abstracts caching from the client (the web application in this case) and is tightly coupled with the database itself.

It is write-through: unlike a CPU-cache, databases are very concerned with data durability, and don't have a way to account for when the results they report to clients (e.g. that a value has been written) are externalized. It is unacceptable much of the time to lose values that were reported as written if power is lost to the database.

Finally, it is generally write-no-allocate, since writes don't use the same names as reads (insert queries vs select queries), though some sub-queries of an insert might be reads themselves, and those might be cached immediately.

The second kind of cache is the one we built in assignment 3, and is a common cache for application to use explicitly (one example is memcache).
In this cache, values are complete application objects (e.g. a user profile, or a constructed HTML page) that may be the result of more than one database query. Keys are generally application meaningful strings that correspond to something a user might request—e.g. a username that might appear in an HTTP query string or a cookie, a slug for a blog post, etc.

These caches are look-aside caches since they are not integrated with a particular database. Instead they leave it to the application to decide how to store or fetch objects from an underlying store—e.g. by performing multiple queries, or a complex join query, for a single key.

The caches are typically used as write-through: an application developer generally shouldn't put something in the cache that isn't synced with the database. There are at least two reasons for this. First, just like with the database cache, web applications generally cannot tolerate lost data after reporting a result back to the user. Second, caches, web servers, and database, often reside on different physical servers, or different processes on the same machine and, as a result, don't fail as a unit. Therefore, it's important to make sure that writes are durable to the database once another thread might see it, even if the cache or web application fail.

Finally, it is up to the application whether to use the cache as write-allocate or write-no-allocate.