Midterms returned

- Mean: 53, std dev = 7
- FYI: HW1 (mean: 51, std = 8) HW2 (mean: 35, std = 5)

Project #2 assignment up late this afternoon

- Implement client functionality of Gnutella (v0.4 protocol)
- Able to connect to Gnutella network; send queries and receive responses
- Transfer content via HTTP
- Don't need to implement flooding etc...

---

Lecture Overview

- How much can caching help performance?
- Web caching
  - HTTP lecture briefly covered cache functionality
  - In this lecture, we go into detail
    » Why do it
    » Where to do it
    » How it performs
- P2P caching
  - How do P2P use distributions differ?
  - How does this impact caching in P2P systems?

---

Why Web Caching?

- Cost
  - Original motivation for adopting caches (esp. internationally)
  - Caching saves bandwidth (bandwidth is expensive)
- Performance
  - User: Reduces latency
    » RTT to cache lower than to server
  - Server: Reduces load
    » Caches filter requests to server
  - Network: Reduces load
    » Requests that hit in the cache do not travel all the way to server

---

Caching in the Web

- Performance is a major concern in the Web
- Proxy caching is one of the most common methods used to improve Web performance
  + Duplicate requests to the same document served from cache
  + Hits reduce latency, bw, network utilization, server load
  + Misses increase latency (extra hops)

---

Where to Cache?

- Answer: Everywhere
  - Browser (user)
    » Small: 1MB memory, 7MB disk (Netscape)
    » Note recursive caching (memory vs. disk)?
    » 20% hit rate
  - Organization (client-side proxy)
    » Large: Gigabytes (with disk)
    » 50% hit rate (for large client populations)
    » In front of server (server-side accelerator)
    » Large (gigabytes)
  - Server itself (in memory)
Proxy Cache Implementations

- Squid proxy cache most popular free cache
- Research project
- Apache web server can be configured as cache
- Many cache products
  - Network Appliance, Inktomi, Infobiqria, etc.

- At this point
  - Web caches are frequently used
  - Issues well understood
- Let’s see how and why they work
- Remember, it’s all about performance

Cache Performance

- Ideally, we want ~100% cache hit rate
  - In practice, we get around 50%
- Cache effectiveness is determined by the workload
- Sharing is the most important aspect of the workload
  - Requests hit in cache because object previously requested
  - Requests to popular objects hit in cache (only first is miss)
- Sharing obeys Zipf’s law
  - # requests \( n \) to an object is inversely proportional to its rank \( r \)
  - \( n = r^{-a} \), where \( a \) is a constant close to 1

Object Popularity

- The implications of the object popularity distribution are interesting
- Cache hit rate grows logarithmically with
  - Cache size
  - Number of users
  - Time
- Easy to get most of the benefit of caching
  - Beginning of the distribution
- Hard to get all
  - Tail of the distribution

Number of Users

- There are a number of reasons why requests miss
  - Compulsory (50%)
    - Object uncacheable (20%)
    - First access to an object (30%)
  - Capacity (<5%)
    - Finite resources (objects evicted, then referenced again)
  - Consistency (10%)
    - Objects change (“../today”) or die (deleted)
Uncacheable Objects

- Caches cannot handle all types of objects
  - Pages constructed from server-side programs
    » “My Yahoo”, E-commerce
  - Changing data
    » Stock quotes, sports scores, page counters
  - Queries
    » Web searches
  - Marked uncacheable
    » Server wants to see requests (e.g., hit counting)

Challenges
- Difficult to solve, not one culprit

Effect of Uncacheability

Approaches to caching more types of web content
- Caching active data: Data sources may be dynamic, but not continuously (e.g., sports scores (Olympic web sites))
  » Snapshots generated from databases
  » Requires cooperation of server and database
- Cache server-side program inputs and outputs
  » Need to recognize program+inputs
- “Active caches”: Run programs (e.g., Java) at caches to produce data
  » Can handle almost anything dynamic
  » Need data sources, though…starts to become distributed server
- Consistency mechanisms (more later)

Prefetching

Let’s say we make everything cacheable
- We still have a high compulsory miss rate (30+%)
- Initial requests to objects
- What to do?
  » We can guess that objects will be requested in future
  » And request them now: prefetch
- Fancy algorithms (markov models with conditional probs.)
  » Simple algorithms (only embedded)
  » Effective: 50% reduction in page latency

Tradeoffs
- Can increase cache hit rate, reduce latency
- But, can be tough to determine what will be accessed
- Accuracy (waste bandwidth), stale data

Cache Capacity

- Caches have finite resources
  » Eventually, something is going to have to be evicted
- Choice is made by the cache replacement algorithm
  » Cache replacement is probably the most popular single web cache research topic
- It also probably has the least impact
  » Capacity misses comprise <5% of miss rate
  » Greatest benefit you could hope for is a 5% improvement
- Basically, want an algorithm incorporating frequency and size
- General problem
  » Fancy algorithms evaluated with small, unrealistic cache sizes
Consistency
- Consistency ensures that objects are not stale
  - Always want version on server and in caches to be the same
- Objects have lifetimes (TTL)
  - Requests to expired objects have to go back to server if Modified-Since (304)
  - If object hasn’t changed, return from cache
  - Otherwise server sends back changed object
- Even if not modified, still suffer extra latency and server load
- TTLs tend to be conservative
  - Shorter TTLs to reduce potential for staleness
  - Results in many requests back to server (10-20%)

Server-Driven Consistency
- Servers know when objects change
- We can have them tell caches when they change
  - Send invalidations
- Leases used to synchronize caches and server
  - Object leases: Short, per-object TTLs
    - Record cache has copy to send invalidations
  - Volume leases: Long, per-site TTLs
    - Amortize lease renewal for many objects
- Key issues
  - State to keep track of objects in proxy caches (can scale)
  - Load induced by bursts of invalidations (pace them)

Cooperative Caching
- Sharing and/or coordination of cache state among multiple Web proxy cache nodes
  - NLANR cache hierarchy most widely known

Cooperative Caching
- Idea: Increase number of users using caching system
  - Have caches “cooperate” and share content, users
  - Caches send their misses to other caches (e.g., to a parent cache in a hierarchy)
  - Can greatly increase number of users in system (and hit rate)
- Cooperative caching has also been a popular topic
  - I’ve even worked on it (part of my thesis)
- Many interesting issues: architecture, request routing, updates, scalability
  - Utility depends on scale
  - Works well for small scales (depts.), but not very necessary
  - Some benefit for medium-scale (large city)
  - Large scale (national) not worth the complexity

Summary of Web Caching
- Web caching
  - Used every step of the way
  - Proxy caches give us about 50% hit rate
  - Many techniques for improving cache effectiveness
  - But cannot be the only answer
- What about P2P systems? Would caching work as well there?

Analysis of Content Distribution (Sariou et al)
- Thirst for data continues to increase (more data & users)
- New types of data have emerged – audio, video
- People use new means to exchange this data
- The result – Internet is now seeing a mixture of old and new content delivery systems:
  - Conventional Web servers and Web clients
  - CDN: Akamai, Digital Island, Speedera
  - P2Ps: Kazaa, Gnutella, Napster, AudioGalaxy
- High-level questions:
  - What is the (bandwidth) impact of these systems on the Internet?
  - What are the characteristics of the new delivery systems?
What applications use the most bandwidth?

- From University of Washington Trace
- Web = 14% of TCP; P2P = 43% of TCP
- P2P now dominates Web in bandwidth consumed!!

Bandwidth use asymmetry...

<table>
<thead>
<tr>
<th></th>
<th>WWW</th>
<th>Kazaa</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>inbound</td>
<td>outbound</td>
</tr>
<tr>
<td>Bytes Xferred</td>
<td>1.51TB</td>
<td>3.02TB</td>
</tr>
<tr>
<td>Unique objects</td>
<td>72,818,997</td>
<td>3,412,647</td>
</tr>
<tr>
<td>Clients</td>
<td>39,285</td>
<td>1,235,108</td>
</tr>
<tr>
<td>Servers</td>
<td>403,087</td>
<td>1,463</td>
</tr>
</tbody>
</table>

Outbound bandwidth usage

- P2P has diurnal cycle, like the Web
- P2P peaks later at night

What kind of data is being sent?

- Web = text + images
- Kazaa = video
- Akamai = images
- Gnutella = video + audio

Object size

- P2P objects are 3 orders of magnitude bigger than Web objects

Most bandwidth consuming object
A few clients and servers exchange big, popular objects

Top 5 bandwidth consuming objects

<table>
<thead>
<tr>
<th></th>
<th>WWW (inbound)</th>
<th>Kazaa (inbound)</th>
<th>Kazaa (outbound)</th>
</tr>
</thead>
<tbody>
<tr>
<td>object size (MB)</td>
<td>object size (MB)</td>
<td># clients</td>
<td># servers</td>
</tr>
<tr>
<td>----------</td>
<td>---------------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>1</td>
<td>0.009</td>
<td>1.4 MB</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>0.052</td>
<td>3 MB</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>2.33</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>0.035</td>
<td>1.4 MB</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>2.23</td>
<td>1.4 MB</td>
<td>14</td>
</tr>
</tbody>
</table>

Object Popularity

1,000 Kazaa objects (out of 111K) responsible for 50% of bytes transferred

How is bandwidth distributed?

<table>
<thead>
<tr>
<th></th>
<th>WWW</th>
<th>Kazaa</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>inbound</td>
<td>outbound</td>
</tr>
<tr>
<td>Bytes Xferred</td>
<td>1.51TB</td>
<td>3.02TB</td>
</tr>
<tr>
<td>Unique objects</td>
<td>72,818,997</td>
<td>3,412,647</td>
</tr>
<tr>
<td>Clients</td>
<td>39,285</td>
<td>1,231,308</td>
</tr>
<tr>
<td>Servers</td>
<td>403,087</td>
<td>1,463</td>
</tr>
</tbody>
</table>

Clients

Small number of clients account for large portion of traffic
- 200 Web+Akamai = 13% of traffic
- 200 Kazaa = 50% of all incoming HTTP

Servers

20 servers supply 80% of Web traffic
- 534 servers (10%) supply 80% of Kazaa traffic

Summary on P2P content

- The Internet is being used in a completely different way:
  - P2P traffic is now the largest bandwidth consumer
  - Peers consume significant bandwidth in both directions
  - P2P objects are 1,000 times larger than Web objects
  - Small number of huge objects are responsible for an enormous fraction of bytes transferred
  - 300 Kazaa objects consumed 5.6TB bandwidth!
  - Few P2P peers are causing much of the traffic
  - Caching isn’t going to help much here…