What is Proxy Caching?
Proxy Caching

- explicit
  - client configuration

- transparent
  - emulate responses coming from origin server
Reverse Proxy Caching

- deploy on “server side”
- increase server capacity
- can have m-m relationship
- hostname of resources resolve to the cache
What is YTS?
YTS Quick Facts (1/2)

- a high performance web caching solution
  - like publicly available Squid, Varnish, HAPerxy, Nginx
- based on Traffic Edge, developed by Inktomi
  - which was acquired by Yahoo! at 2002
- is a multi-threaded state machine
  - scales very well on modern multi-core box
  - fully leverages multi-core CPUs
YTS Quick Facts (2/2)

• YTS is a 32-bit application
  • but we recommend running YTS on 64-bit OS
• includes a fast, asynchronous DNS resolver
  • directly issuing DNS command packets
• supports an extensible plug-in architecture
  • with a large list of APIs for modifying requests or responses
  • allow property or platform needed custom behavior
    – YMail gateway
    – Wretch IAV reverse proxy
    – could act as SW L7, replace HW load balancer
YTS v.s. Squid (1/3)

- **hardware**
  - 2 * Xeon E5320 1.86GHz
  - 8GB DDR2-667 ECC fully buffered
  - 6 * 147GB 15K SAS/3

- **benchmark**
  - variable cache hit ratio percentages (0, 50, 95, 100)
  - 1,000 client connections
  - 1KB response from the origin
  - 4 keep-alive requests per connection
  - 10,000 unique objects
YTS v.s. Squid (2/3)

- The graph illustrates the comparison between YTS and Squid in terms of cache hit rate and requests per second.
- The red line represents YTS, showing an increasing trend as Cache Hit Rate increases from 0 to 100.
- The green line represents Squid, with a more modest increase in requests per second as Cache Hit Rate increases.

This suggests that YTS scales better under high cache hit rates compared to Squid.
YTS v.s. Squid (3/3)

The graph compares the Mean Response Time (in ms) against the Cache Hit Rate (%) for YTS and Squid. The green line represents YTS, showing a decreasing trend as the Cache Hit Rate increases, indicating improved performance with higher hit rates. The red line represents Squid, following a similar trend but with a different slope, suggesting a different performance characteristic compared to YTS.
How Yahoo! use YTS?
Yahoo CDN: based on YTS

- at November 2009
  - 128TB per day
  - 17 billion requests per day
  - peak at 20+Gbps
  - targets 90% of content being cacheable
Yahoo Connection Proxy (1/3)

- reverse proxy of connection to end users
  - located at last mile
- reduced round-trip times
Yahoo Connection Proxy (2/3)

- provides keep-alive support
  - a connection is re-used for subsequent objects being downloaded
  - TCP slow start will only impact the first object downloaded
Yahoo Connection Proxy (3/3)

• allowing for domain collapsing
Open Source!

- Yahoo! donate the source to Apache Software Foundation at Nov/2009
Setup YTS
Quick Setup a Reverse Proxy by YTS

- install YTS
- creating mapping rules (remap.config)
  - map http://tw.image.XXX.yahoo.com/oa  http://tw.m.XXX.yahoo.com
- setup cache storage (storage.config)
- enable reverse proxy (records.config)
- DNS entry of the advertised hostname of the origin server => YTS
  - tw.image.XXX.yahoo.com => tw.ytsvip.XXX.yahoo.com
Hierarchical Caching

- parent, sibling, neighbor/peer relationships
- using Internet Cache Protocol (ICP)
- “neighbor miss” – request forward to a parent or directly to origin server
- each node cache the object on miss
Cluster Caching

- management only
  - nodes automatically share configuration
- full clustering
  - include management-only mode
  - cache is distributed across nodes into a single, virtual store, rather than replicating the cache node by node
  - enormous aggregate cache size & maximize hit rate
  - recommended to use a dedicated NIC for cluster communication
- use a proprietary communication protocol
Object Freshness

- by expires & max-age header
- by formula
  - freshness_limit = (date – last_modified) * threshold
- absolute limit
- revalidate rules in cache.config
  - for particular domains / IPs / regular expressions, etc
Cache Management

- cache preloading
  - scheduling updates
  - immediate updates
  - pushing via HTTP PUSH
- cache pinning
Caching Special Objects

- caching dynamic content
  - URI with “?” “;” “cgi” or ends in “.asp”
- caching cookie-d objects
- caching HTTP alternates
  - different objects with the same URL
Determining the Cache Size

- related to system’s temporal locality
- cache size need = (average object size) * (# of cached objects to achieve our target hit rate)
- for example, to achieve 98% hourly hit rate,
  - average object size ~ 35kb
  - # of cache objects required ~ 800k
  - cache required: 800k * 35kb = 28GB
Disk Cache

- setup in storage.config
- cooked disk
  - 64MB cache storage in /tmp/cache.db
    - /tmp 67,108,864
- raw disk
  - skips the OS I/O buffer (preferred in Yahoo!)
  - use the '/usr/bin/raw' to bind a raw device to an existing block device
    - e.g. raw /dev/raw/raw1 /dev/sys1/raw
- 72GB raw disk cache
  - /dev/raw/raw1 77309411328
- don’t set the disk cache much larger than what you need
Ram Cache

- setup in records.config
  - proxy.config.cache.ram_cache.size
- current YTS is 32-bit only
  - max 2.5-3GB of ram cache on 64-bit OS
- memory used for cache = ram cache + in memory indices for disk cache
  - the indices is linear in size to the disk cache configured
  - e.g 500GB /w 700MB memory indices
- defaults is to use 1MB memory per GB disk cache
- YTS will crash if the ram cache size is set too high
Other Optimizations

- maximum size of objects allowed in RAM cache
  - proxy.config.cache.ram_cache_cutoff
  - eg: 35KB

- maximum size of objects allowed in cache
  - proxy.config.cache.max_doc_size
  - eg: 32MB
Miscellaneous

• YTS startup
  • it can take several seconds (even up to 10s) until the cache is initialized
  • no cache hits at this period
  • better have the machine out of rotation until YTS finish initialization

• object with “different” URLs
  • http://tw.XXX.yahoo.com/test.jpg
  • http://tw.XXX.yahoo.com/test.jpg?.r=123 (default is to cache dynamic)

• customization
  • plug-in architecture - allowing properties to modify YTS's behavior when necessary
YTS Plug-in Example

• regular expression mappings
• WoE (geographic) based ACL
• cookie based routing
• OAuth request caching
• map CAPTCHA session
• traffic throttle
Conclusion
Putting It All Together

User connects to site.yahoo.com at the closest edge POD

Keep-Alive between Remote And Local

Static content (images, css, js)
from the closest edge POD

UGC Images under /photos/

L7 URL Hashing

YTS Cache Servers

Property servers for dynamic (PHP) generated pages

Origin Servers
More Information

• http://trafficserver.apache.org/docs/
• http://trafficserver.apache.org/docs/v2/admin/
• http://svn.apache.org/viewvc/trafficserver/
• https://cwiki.apache.org/TS/faq.html
Question ?