Web Proxy Caching

1 introduction

1.1 motivation

- web is slow
- the same data is sent to many people

1.2 brief traffic discussion

- temporal locality of requests
- consecutive requests often go to the same server
- ...

1.3 introduce and define web caching

- web objects
- browser cache
- proxy cache
- server cache

1.4 contrast web cache styles

- where in network
- what issues are addressed
- what resources are addressed
  - what bottleneck is each type of cache dealing with
- contrast with other caches
  - memory cache is well known
  - variable load time
  - variable size
  - source of data changes

1.5 why focus on proxy caches?

- proxy caches are a significant research area
- also a significant area of commercial development
- advantages of proxy caches
- disadvantages of proxy caches

2 how a proxy cache works

2.1 discuss at high level

2.2 include basic state diagram for proxy cache

2.3 outline issues within this context

2.4 pull out our 7/8/9 issues

3 issues in design

3.1 quantifying performance

- the importance of the evaluation of performance
- performance evaluation metrics
  - web cache hit ratio
  - byte hit ratio
- performance issues as design choices

3.2 cache consistency

- What is the purpose of cache consistency mechanism ?
  - mechanism ensuring cached copies of data are eventually updated to reflect changes to the original data
- What makes the web cache consistency mechanism different from traditional ones ?
  - large population of caches
  - long latency
- What is the basic HTTP mechanism to support cache consistency ?
  - HTTP headers
- What is the metric for the degree of consistency?
  - staleness/ freshness
- What is the consistency level ?
  - strong consistency
  - weak consistency
- approaches for cache consistency
  - TTL-based
  - invalidation-based
  - lease-based

3.3 replacement policies

- why we use replacement policies

  First,recent studies has shown that Web cache hit ratio (HR) and byte hit ratio (BHR) grow in a log-like fashion as a function of cache size. Thus a better algorithm that increases hit ratios by only several percentage points would be equivalent to a several-fold increase i cache size.

  Second, the growth rate of Web content is much higher than the rate with which memory sizes for Web caches are likely to grow. The only way to bridge this widening gap is through efficient cache management.

  Finally, the benefit of even a slight improvement in cache performance may have an appreciable effect on network traffic, especially when such gains are compounded through a hierarchy of caches.

- existing replacement algorithms According to whether they exploit access recency or access frequency, the existing replacement algorithms can be classifies into three categories.

  - Recency-Based Policies. LRU is the most widely used cache replacement algorithm,as it captures recency and is superior to other policies.

    the disadvantages of LRU

  - Frequency-Based Policies.

    The basic frequency-aware replacement algorithm is LFU.

    There are two subtle problems with LFU.

    First,there are different versions of LFU algorithm.

    Second, in an LFU replacement algorithm, when two objects have the same reference count, a tiebreaker is necessary.

  - Recency/Frequency-based Policies.

    (1) Several studies have considered both recency and frequency information under a fixed cost/fixed size assumption.

    a.The LRU-K algorithm maintains the last K reference times to each object to compute the avrage reference rate.

    b.The LFU-DA algorithm is a frequency-based algorithm with dynamic aging. c.Another work generalizes LRU and LFU to a hybrid policy LRFU

    (2)To deal with variable cost/size,generalizations of above techniques have also been proposed.

    a. the LNC-W3 algorithm is proposed as a generalization of LRU-K to deal with variable-cost and variable-size Web objects.

    b. in another direction several studies proposed generalizations of the GreedyDual-Size algorithm to incorporate frequency. These algorithms include GDSF and GD-LFU.

3.4 prefetching

- motivation
  - why prefetching is needed
- limitations
  - bounds of latency reductions from prefetching
- side effects
  - increased network traffic
  - increased server load
- classification of prefetching techniques
  - by location
    * between proxy cache and server
    * between browser cache and proxy cache
  - by knowledge resource
    * server hints - popularity information
    * local hints
      · single user history
      · multi-user history / local popularity
    * combined server and local information
    * web page itself
- by content
  - content prefetching
  - connection prefetching
- bandwidth smoothing
- prediction algorithms
  - partial matching prediction
  - statistical methods - using popularity information
  - interactive prefetching - study the HTML and links
  - access tree
  - data mining (web mining) - event sequence prediction
  - resource grouping
  - connection prefetching: dns, tcp, dummy headers

3.5 cachability

- What kind of docments are usually considered uncacheable
  - requests requiring authentication
  - session-encrypted data
  - rapidly changing documents
  - very large documents
- making some of the cachable
  - documents with some type of cookie can be cached
  - pseudo-dynamic data might be cached
    - a dynamic composition or stored or static data
- techniques
  - for documents with cookies
  - for pseudo-dynamic data
  - active caching
  - stream caching

3.6 architecture

- disk vs. memory based caching
  - size considerations
  - proxy vs. server vs. browser
- distributed / parallel caching
  - motivation
    - proximity to clients
    - size benefits
  - handling requests
    - url routing
    - load balancing requests
  - parallel synchronization issues
    - internal coherence
    - load balancing cache contents
    - replication of data
  - parallel architectures
    - peer-to-peer
    - hierarchical
- different data flow models

3.7 security and privacy

- activity tracking
- fake content
- document modifications

3.8 reverse caching (?)

3.9 stream caching (?)

4 Performance Measurement

- existing evaluation methods
  - The space of cache evalution methods
  - Methodological Appraisal
    * Simulated systems
    * Real systems/isolated networks Real systems and networks
    * Artificial workloads
    * Captured logs
    * Current request stream workloads
  - Placement of current work in the space of methodologies
  - Area A1:Simulations using artificial workloads
  - Area A2:Simulations using captured logs
  - Area A3:Simulation based on current requests
  - Area B1:Real systems on an isolated network using an artificial dataset
  - Area B2:Real systems on an isolated network using captured trace logs
  - Area B3:Real systems on an isolated nextwork using current requests
  - Area C1:Real systems on a live network using an artificial dataset
  - Area C2:Real systems on a live network using captured logs
- implications of design choices for performance

5 case studies

- Squid
  - include results from Brown CS cache
- Japan Cache Project
- Akamai
- ...

6 conclusion