

# Optimizing Object Freshness Controls in Web Caches

Mark Nottingham

*mnot@pobox.com*

## Introduction

- Subject is well-established in available tools, but not well understood or used effectively
- Started with Squij, getting more general
- Very much a WIP

## What is Refresh Control?

- Definitive freshness data is not always available from Web servers, because it often requires manual intervention (e.g., setting Expires headers)
- Refresh Control is used by caches to simulate freshness data, based on object characteristics and cache administrator choices

## Benefits of Refresh Control

- Primary: reduce latency by avoiding validation
- Secondary: reduce bandwidth by artificially extending freshness

## Risks of Refresh Control

- Serving stale objects

benefits

## Refresh Control Mechanisms

- Must be based on information available to the cache
- Often use some criteria for classification of objects, allowing fine tuning

mechanism

## Current Practice – Control Mechanisms

- Regex applied to request URL
- Controls:
  - min freshness time
  - Last-Modified percent
  - max freshness time
- LM percentage allows freshness calculation based on object freshness history

## Current Practice – Utilisation

- Informal survey (Squid-Users list)
- Many use default pattern – (Squid: 20%)
- Others use published control lists (what are these based upon?)
- Majority just guess
- All of these risk either serving stale or not seeing potential latency gains

## How Can We Arrive at More Useful Controls?

- Research attributes of common objects, to recommend classifications as well as default settings.
- Develop tools to aid in determining appropriate settings (possibly active).

## Relevant Work – Object Characteristics

- Object classifications - by MIME type, extension
- Object freshness characteristics - age by type
- Object access characteristics

related work

## Trace-Driven Analysis

- Simulate a cache and compare decisions about freshness to what the trace says
- Basic metrics for each refresh control:
  - True/false fresh hits
  - True/false stale hits
- To find patterns in accesses/object characteristics that we can exploit

trace

## Trace-Driven Analysis – Finding a Trace

- Need object validators (Last-Modified)
- Must be non-cached trace
- Time span of trace very important
- Number of accesses not so important
- Two candidates: UCB Home-IP and DEC

trace

## Trace-Driven Analysis – Results

- Period of longest appropriate trace (DEC) too short - not enough stale hits in cache
- Can see the effects of the refresh controls with different settings
- Need better traces

trace

## One Possibility: Traffic-Driven Optimization

- Using cache logs, look at:
  - fresh vs. stale
  - for fresh hits: modified vs. unmodified on the server
- mod/unmod correlates to Fs/Ts in trace
- Do this for each refresh control classified
- Apply results to optimize refresh controls

## Experimental Implementation – Squij

- Gives statistics for each classification:
  - average service time
  - hit rate (raw hits/bytes)
  - for hits, fresh/stale ratio
  - for stale, modified/unmodified
  - amount of traffic (raw hits/bytes)
- Good subjective results so far

## Conclusions / Further Work

- Long-term (6 month+), non-cached traces are needed
- Once validated, fine-tuned default controls can be distributed with caches
- active, traffic-based tools may be viable, if relationships can be established based on data available in cache logs