Prequel: Distributed Caching for the Masses

Christopher R. Hertel
Senior Principal Software Engineer, Red Hat

José A. Rivera
Software Engineer, Red Hat
Introductions
Introductions

Christopher R. Hertel  
Roa Perez Amezaga

Your friendly neighborhood SMB/CIFS geek.

Swimming in the deep end of the SMB/CIFS pool since 2008.

Mostly Harmless.
The opinions expressed are our own and not necessarily those of our employer, our respective spouses, pets, or "the Voices".
Where are we going?
...and what am I doing in this handbasket?

- BranchCache Overview
- The Prequel Project
  - PrequelID
  - PrequelHC
- Client Plans
- Tools
BranchCache Overview
BranchCache Overview

Accessing content over a WAN link
- Minimize content copies over the WAN
  - Cache the copy on the local network
- Ensure that the cached copy is still valid
  - Retrieve fingerprints from the server
Clients request “fingerprints”
- Each fingerprint maps to a chunk of content
- Fingerprints are used to find cached content
- If content is not found in the local cache, it is retrieved over the WAN
- Cache keeps fingerprint-to-content mapping
Branch Cache Overview

Distributed Cache Mode:
- Each node keeps a cache of content it has downloaded
- Clients broadcast to find content
- The cache is distributed across peers
- Limited to the local LAN
Hosted Cache Mode:
- Clients tell cache node that they have cache-able content
- Cache node retrieves cache-able content from the client node
- Other clients always query the cache node for content
- Not LAN-locked
Any questions about BranchCache basics?
Prequel Project
The Prequel Project

- An Open Source PeerDist implementation
- PeerDist is the protocol suite underlying BranchCache
Prequel Project

Prequel Project Goals

- PrequelD: Server-side hash generation
  Interface with:
  - Samba
  - HTTP server (e.g. Apache)
- PrequelHC: Hosted Cache
- Prequel Client for Linux
- Prequel Tools
Prequel Project

Websites:

- [http://fedorahosted.org/prequel/](http://fedorahosted.org/prequel/) Source code repository
- [http://ubiqx.org/proj/Prequel/](http://ubiqx.org/proj/Prequel/) Project home page

Microsoft Docs:

- [MS-CCROD] Content Caching and Retrieval Protocols Overview
- [MS-PCCRC] Peer Content Caching and Retrieval: Content Identification
PrequelD: the Prequel Server Daemon
PrequelD is a Userland Dæmon

- Make “nice”
- Background hash generation
- Hashes stored in cache files
- Cache files are “shared read”
- Speak to Dæmon over a socket
- Threaded communication
PrequelD

Currently “works”

- Needs signal handling
  - SIGHUP: Reload Config
  - SIGTERM: Clean shutdown
- Should traverse directories in the background (feature)
- Should do stale cache cleanup

Cache File Access

- API defined
- Code should be done soon

Supports only PeerDist v1

Design allows for PeerDist v2
Prequeld

Configuration File

```plaintext
socket /var/run/prequeld.sock;
logfile /var/log/prequel.log;

cachedir /var/prequel_cache
  {
    hash1 sha256;       # Most common hash type
    hash2 none;
    keyfile /etc/prequel/prequeld.key;
    sourcedir /data/music
      {
        keyfile /etc/prequel/music.key;
        minblocks 4;
        verbosity 0;
        exclude *.tmp, /Queen/*;
      }
  }
```
Configuration File

- Scoped configuration
  - Global section
    - Most settings have reasonable defaults
  - cachedir sections
    - Identify the target directory
    - Contained settings are section local
    - Settings apply to sourcedirs that follow
    - Order is important!
- sourcedir lines or sections
- Identify directories of source files
- Comments start with a '#'
Design Challenges

- The hash cache is not directly connected to the source content
- Cache files can get out of sync
- May need to be re-hashed
- Hashes should be removed on source file write or delete
- Cache files can become “orphans”

Would it be better to keep the cache within the file system?
PrequelHC: the Prequel Hosted Cache Server
PrequelHC:
the Prequel Hosted Cache Server

(Coming Soon)
PrequelHC

- Stand-alone HTTP server
- Implements two sub-protocols:
  - PeerDist Hosted Cache Protocol ([MS-PCHC])
  - PeerDist Retrieval Protocol ([MS-PCCRR])
- Written in Python
Hosted Cache Protocol

- Used by clients to offer content to the hosted cache server
- Used by servers to fetch content information from clients
- PeerDistv1 requires HTTPS
- PeerDistv2 requires HTTP
Retrieval Protocol

- Used by hosted cache server to fetch offered content from clients
- Transmitted over HTTP
- Data blocks are encrypted over the wire
THE FUTURE

- C libraries for sub-protocols
- Apache module? CGI script?
- Maintain stand-alone server?
Prequel Client: the Uncharted Territory
A user-land client would be fairly easy
- Simple user management
- Applications would need to call it directly

An in-kernel client is more daunting
- Could integrate with the "CIFS" client
- Could mesh with the file system cache
- Available to sync with de-duplication
Prequel Tools:
Catch as Catch Can
Prequel Tools

Tools we've slapped together as we build and test our implementation.

- **PdDump**
  PeerDist v1 Content Information Dump

- **pq_size_calc**
  Calculate the Content Information size from the original file size

- **oSSL_key_dx**
  Decrypt a BranchCache key extracted from Windows
Prequel Tools

Tools we've slapped together as we build and test our implementation.

- **STiB**
  Retrieve Content Information over HTTP (also implements BITS protocol)

- **pq_cgi**
  CGI program generates Content Information on the fly
The End
Prequel Project

 Websites:

  - http://fedorahosted.org/prequel/
    Source code repository
  - http://ubiqx.org/proj/Prequel/
    Project home page

 Microsoft Docs:

  - [MS-CCROD] Content Caching and Retrieval Protocols
    Overview
  - [MS-PCCRC] Peer Content Caching and Retrieval:
    Content Identification
  - [MS-PCHC] Peer Content Caching and Retrieval:
    Hosted Cache Protocol Specification