Implementing PeerDist:

The BranchCache™ Protocol

Christopher R. Hertel
Samba Geek
May, 2012

Implementing PeerDist:

The BranchCache™ Protocol

Christopher R. Hertel
Samba Geek

May, 2012

Terms of Uselessness

Acceptance of Terms. By viewing the content of this presentation, you acknowledge that you have read and agree to abide by the terms described below. If you do not agree to the terms presented here, you must discontinue use of personal hygiene products and staple a walrus to your forehead.

Limitation of Liability. You agree that, whatever it is, if it is bad it is not my fault, it was never my fault, and if anyone says it is my fault you will be very, very angry with them for saying such a mean and nasty thing about me. Further, you agree that if it is good it must be mine and that I am owed a lot of money by a lot of people for making it good. This includes, but is not limited to, ice cream, literature, the space-time continuum, and personal hygiene.

Intellectual Property. Any thoughts or ideas which may occur to you while viewing the content of this presentation are belong to me. You are required to submit scans of brain activity for the period of time during which you viewed this presentation so that the relevant brain cells can be extracted, at your expense.

Your Rights. By viewing the content of this presentation, you agree to forfeit all rights and privileges for the greater good of the American Corporate Oligarchy, particularly if you are not a citizen of the United States and may, therefore, still have rights.

The terms of this agreement are foolish and shall not be enforced.

Introductions

Intermissions

Me

If you don't know me by now...

Introspections

Now with

redhat®

The opinions expressed are my own and not necessarily those of my employer, my spouse, my (lack of) religion, or my dog.

Introductorations

You

Introdictations

Cast of Characters

BranchCache: *The Product*

PeerDist: *The Protocol*

Prequel: *The FOSS Implementation*

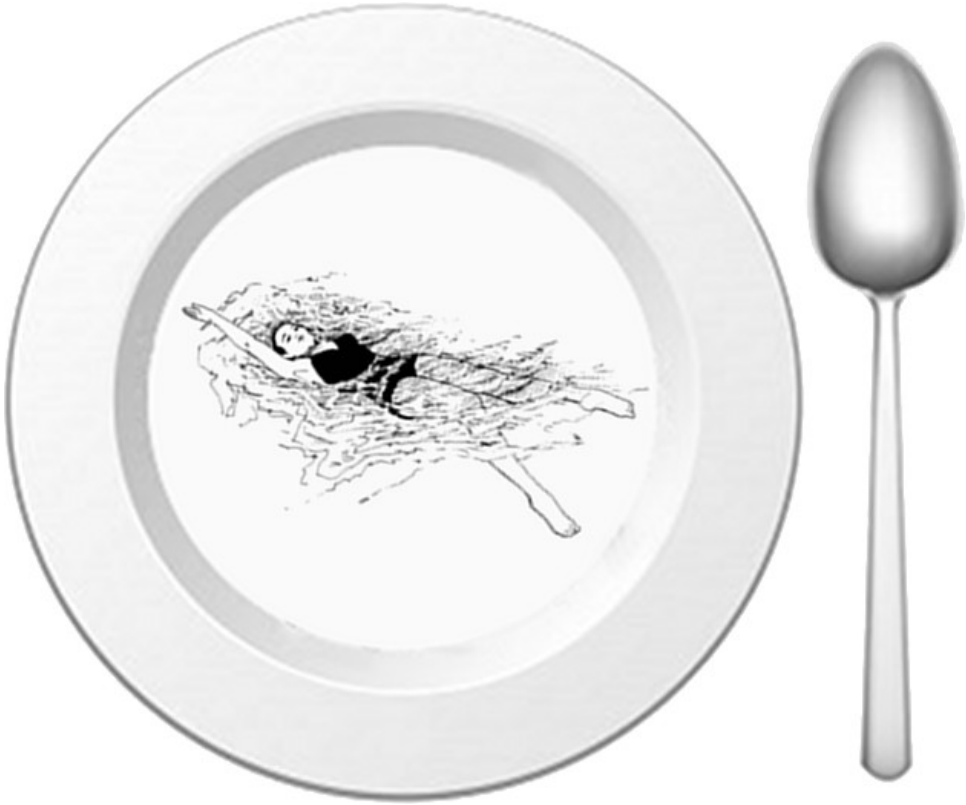
Interrogations

Where Are We Going?

- What is BranchCache™ and why do we want it?
  - PeerDist over HTTP
  - PeerDist over SMB2 and SMB3
- What is the Prequel Project?
  - Prequel CGI
  - Prequel Dæmon
  - Prequel in Linux
- Where do we want to be tomorrow?

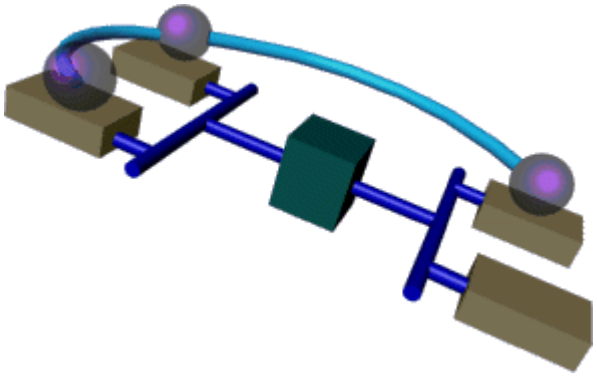
What is BranchCache™?

...and what is it doing in my soup?



What is BranchCache™?

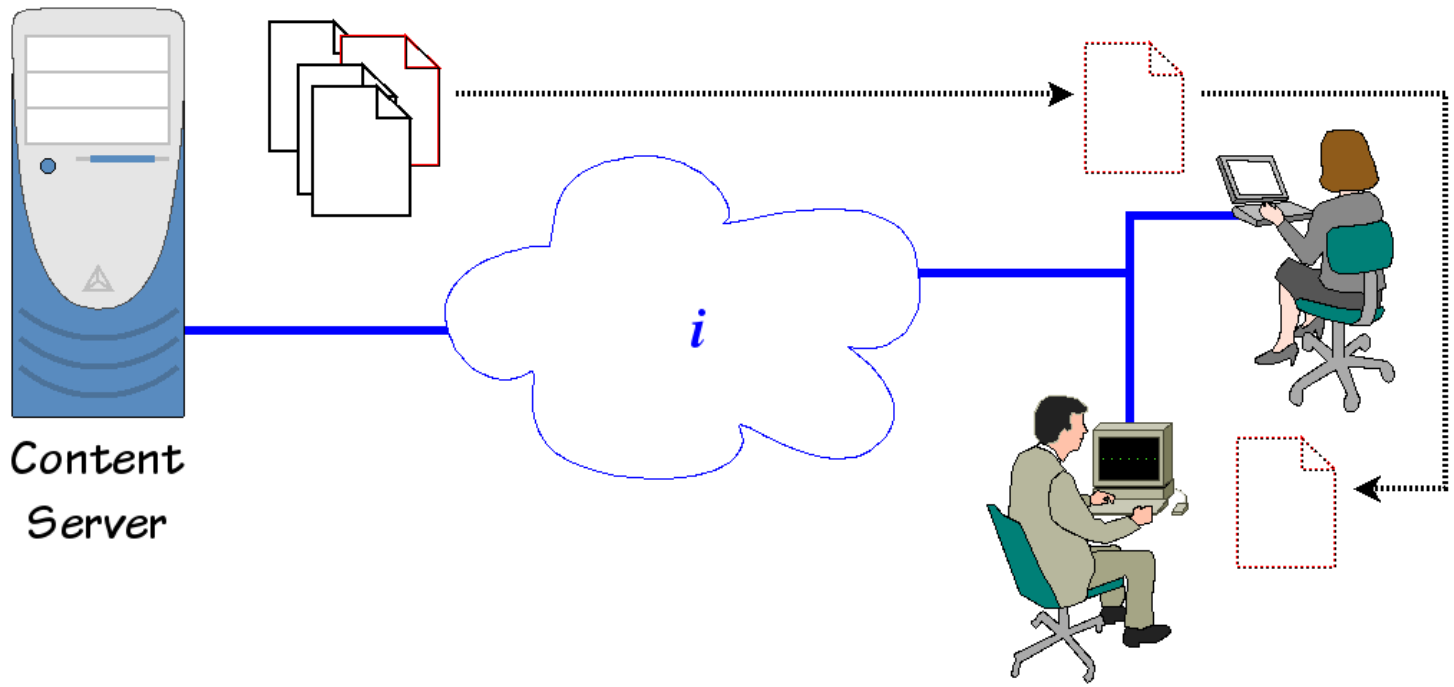
BranchCache™



A Distributed File Caching System

- Basic, simple WAN acceleration
- Supported in SMB2.1+, and over HTTP

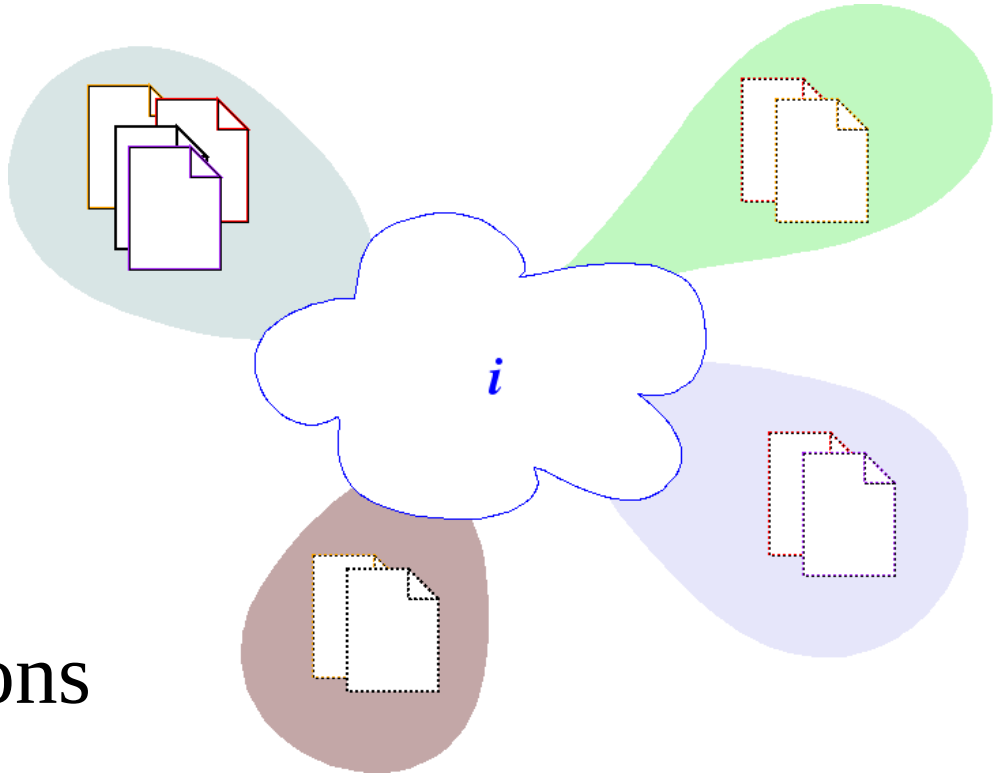
What is BranchCache™?



The basic idea:

- The 1st user fetches a file from the remote server
  - A copy is kept somewhere on the local LAN
- The 2nd user fetches the copy, not the original
  - Local transfer is faster and cheaper

What is BranchCache™?



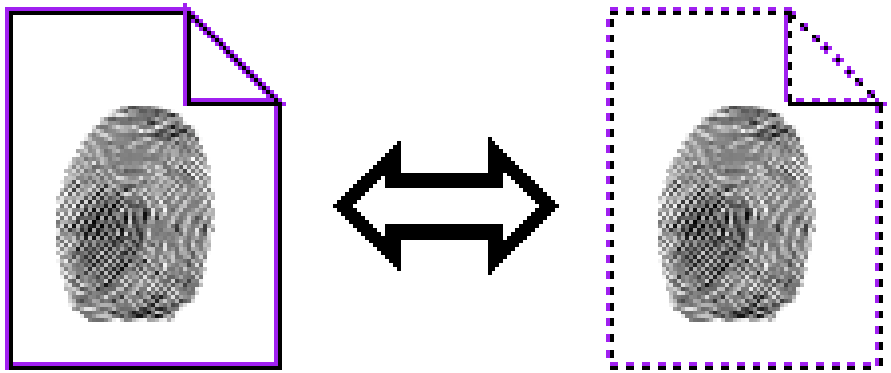
Cached Copies
May Exist in
Several Locations

- Copies are stored on the local networks
  - Copies are *not authoritative*
- What happens when the original changes?
  - How do clients detect the change?

What is BranchCache™?

Before the client can use the copy

- A file fingerprint must be retrieved from the server
- The fingerprint is much smaller than the original file (≈1/2000th)



If the **cached fingerprint** does not match the **server fingerprint**, the client must re-fetch the data.

What is BranchCache™?

Is This Cool?

Simple, easy to implement, *de facto* standard WAN Acceleration.

That's cool...but can we do more with it?

- Distributed file system?
- Scatter-gather network backup?

What is BranchCache™?

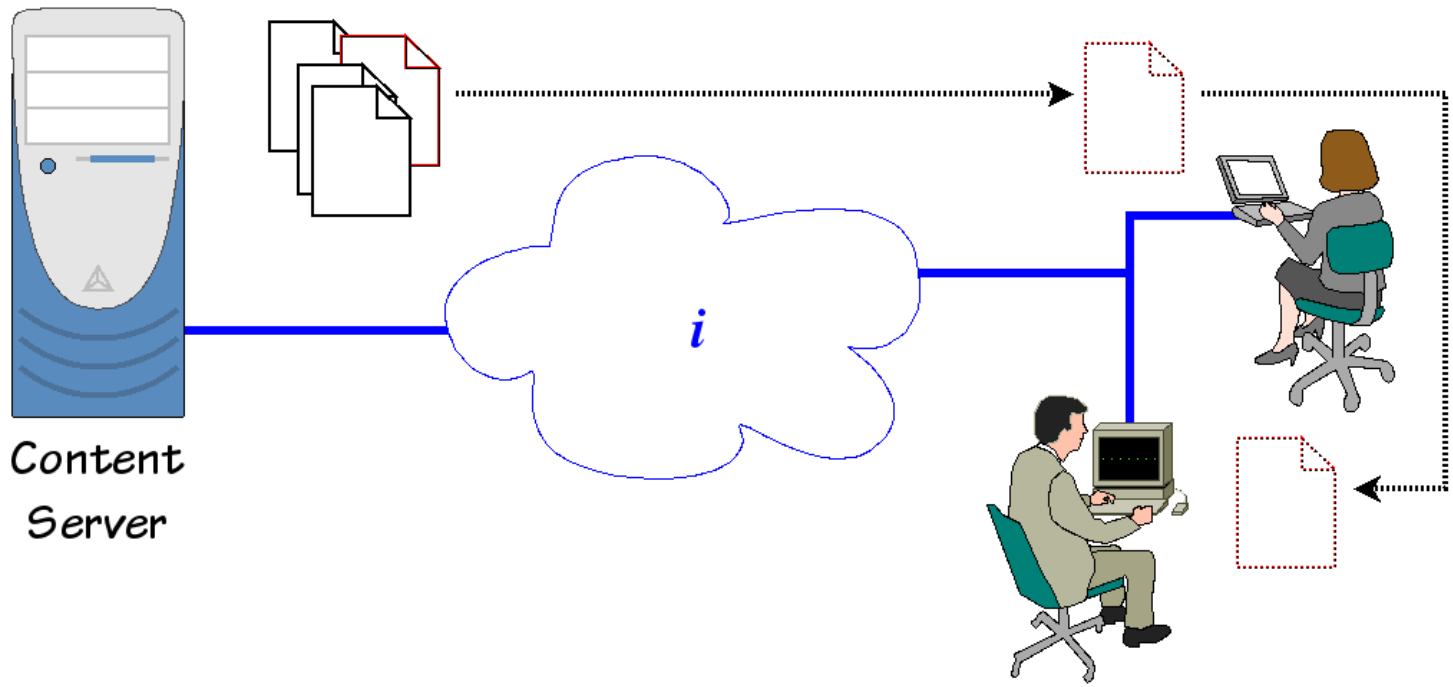
How this is done: the details

BranchCache is an implementation of PeerDist

PeerDist is a documented Microsoft protocol:

- [MS-CCRSO] – Overview
- [MS-PCCRC] – The 'fingerprint' format
- [MS-PPCRD] – Finding local copies
- [MS-PCCRR] – Retrieving local copies
- [MS-PCCRTP] – PeerDist over HTTP
- [MS-PCHC] – Using a hosted cache
- [MS-SMB2] – PeerDist over SMB2.1+

What is BranchCache™?

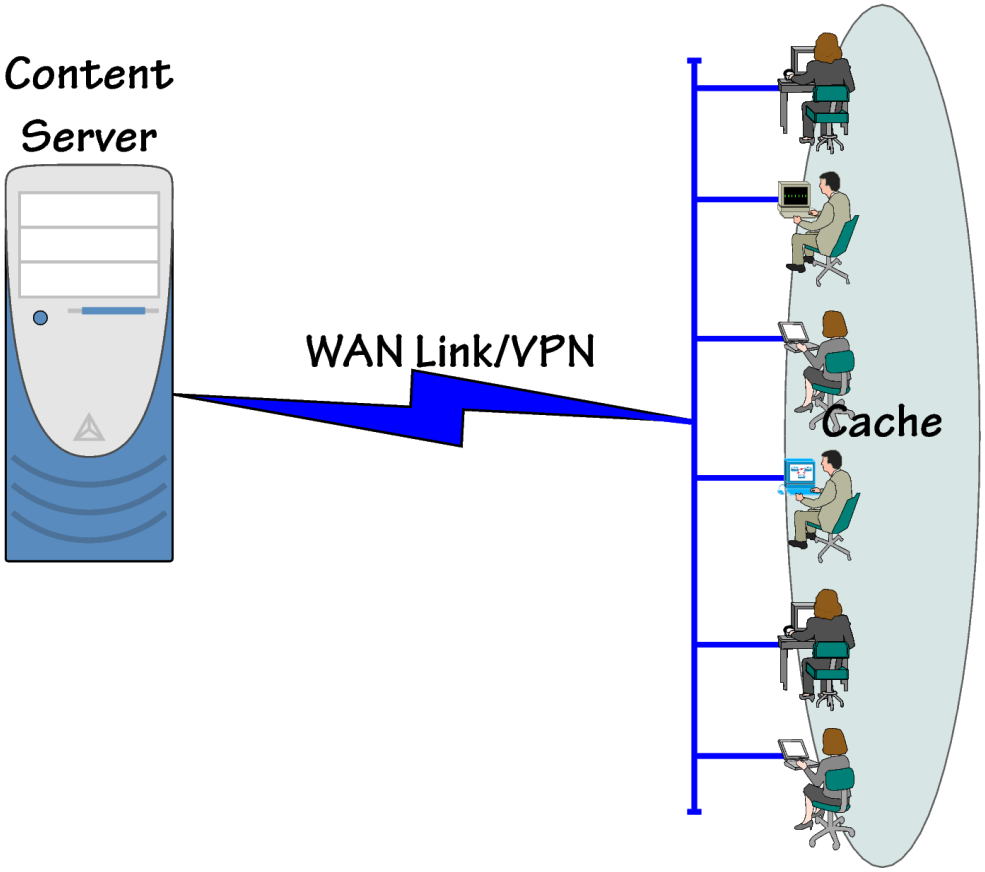


Remember these folks?

- The 1st request tries to get both the fingerprints and the content.
  - Both are required in order to cache the content.
- Once the data is cached, subsequent client requests need only retrieve the fingerprints.

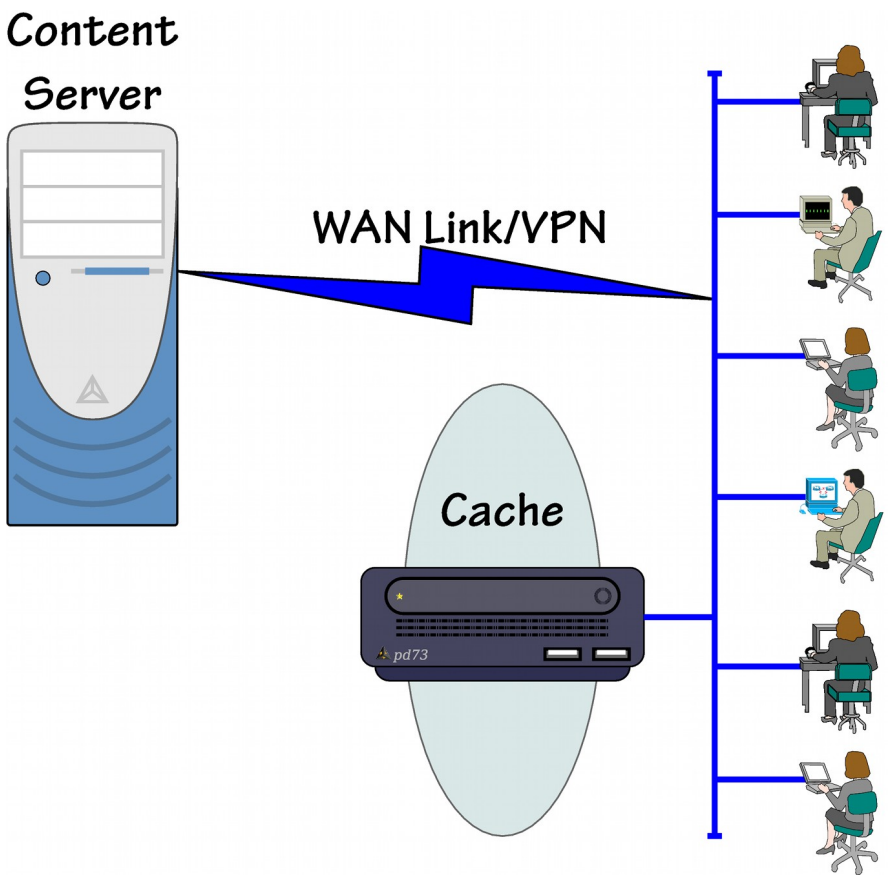
What is BranchCache™?

PeerDist: Peer-to-Peer mode



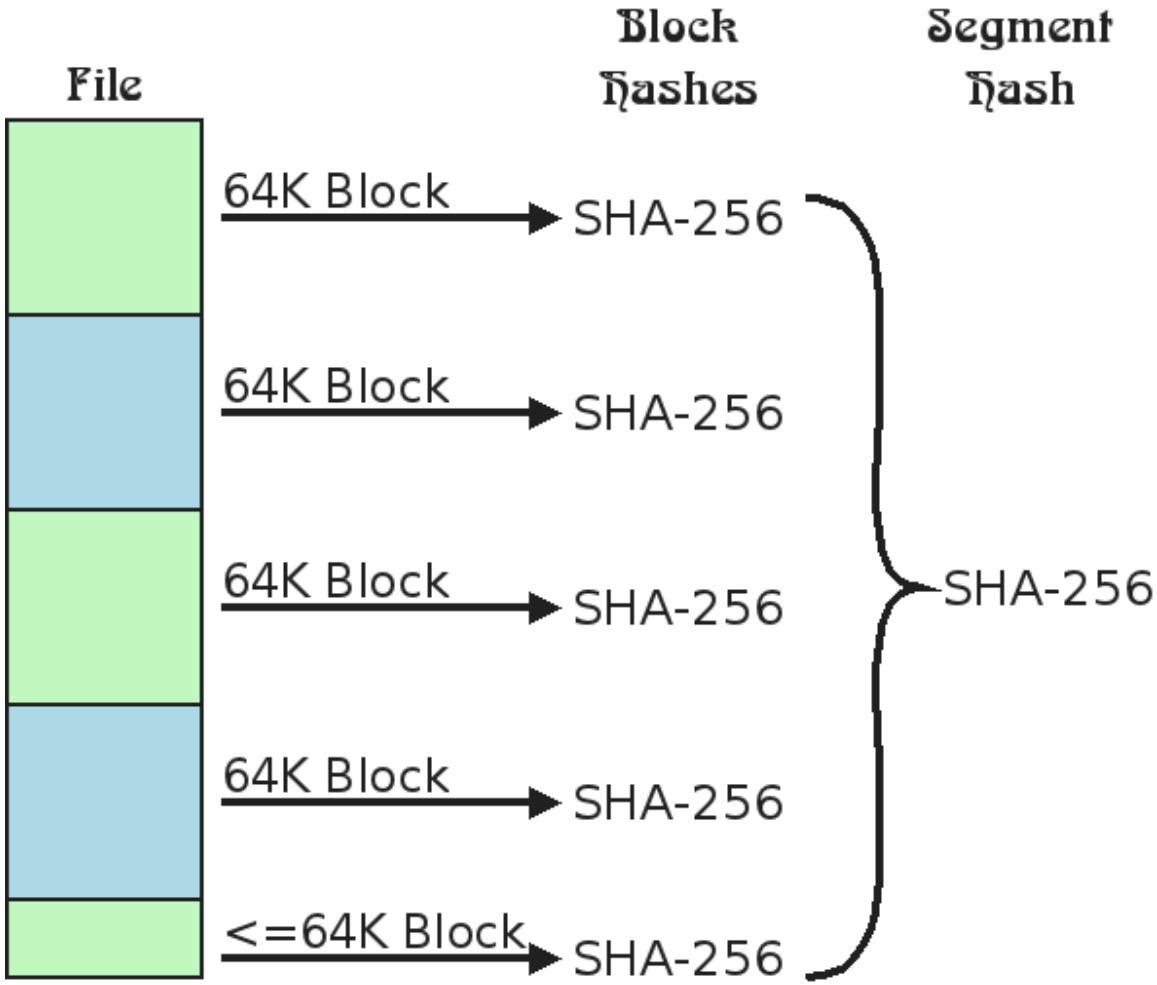
What is BranchCache™?

PeerDist: Hosted Cache mode



What is BranchCache™?

PeerDist v1 (Windows 7)



What is BranchCache™?

PeerDist v1 (Windows 7)

- Each 64K block is hashed
  - The last block may be less than 64K
- Every 512 block hashes (representing a 32M segment) are hashed
  - The last segment may be less than 512 blocks
- Each Segment Hash is HMAC signed using a server secret key
  - The signing key can be extracted

What is BranchCache™?

PeerDist v2 (Windows 8)

Needs more study

- The format is different
- The fingerprinting algorithm is unspecified, leaving “Innovation Space” for implementers

A chance to leverage other Linux development efforts (e.g., BtrFS deduplication).

Prequel

Prequel

Prequel: A small project to implement PeerDist for Linux.

Early success

- `pq_cgi`: A CGI program for HTTP servers that can create PeerDist v1 Content Information[1]
- The structure of the PeerDist v1 fingerprint does not lend itself to on-the-fly delivery

[1] Content Information == Fingerprint

Prequel

Prequel: A small project to implement PeerDist for Linux.

Some Simple Tools

- `pdDump`: Dumps PeerDist v1 Content Information
- Tools to extract a Windows BranchCache Server Secret (signing key) and Passphrase

These are proof of concept tools, and they do prove the concept.

Prequel

Prequel Dæmon

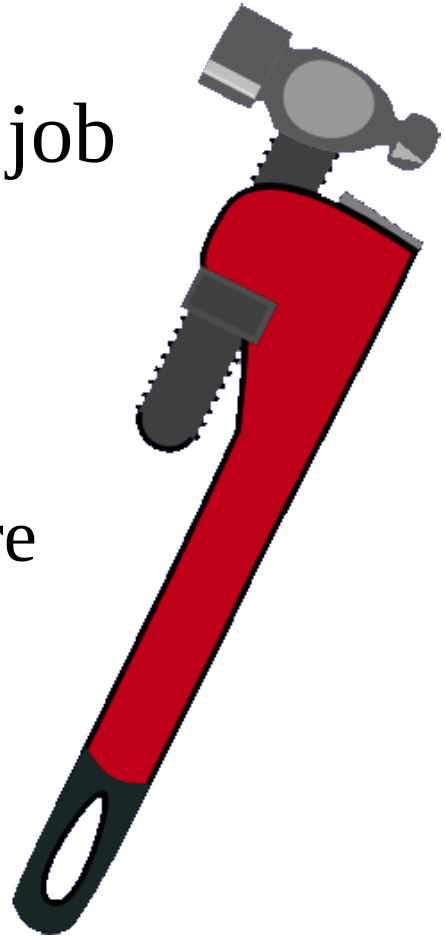
Work in Progress

- `PrequelD`: Generate Content Information files from original data
- Make Content Information available to HTTP servers and Samba
- Garbage-collect outdated Content Information files

Prequel Dæmon

Possibly not the best tool for the job

- Access rights complexities
- Must scan source files to detect changes and new files
- Needs separate directories to store Content Information files
- Must check Content Information files to ensure that they are still valid



Prequel

The Dream: Kernel PeerDist

Content Information as FS metadata

- Content Information generated asynchronously
  - invalidated on write to file
  - accessed using an open file handle
- Kernel lease (OpLock) handling is correct
- Children sing, Flowers bloom, Sun shines

Prequel

The Dream: Kernel PeerDist

A few outstanding questions...

- Can this be done in a kernel VFS?
  - Where would Content Information be stored?
- Can we access BtrFS de-dupe tags for use as a PeerDist v2 fingerprint?

Prequel Client-side Cache

PeerDist Consumers

- Clients need to be PeerDist aware
  - They must know to request PeerDist
  - They must know how to retrieve content from the cache

- Where should the cache be kept?
  - In the kernel or in userspace?

Prequel Client-side Cache

PeerDist Consumers

- Top consumers will likely be the CIFS kernel client and user-land HTTP clients (including package update programs)
- Both peer-to-peer and hosted mode should be supported by clients

The Prequel Project

Getting our act together

- Homepage:
  http://www.ubiqx.org/proj/Prequel/

- Wiki and Git (just recently created):
  http://fedorahosted.org/prequel/

The End