LA-UR-21-32308

Approved for public release; distribution is unlimited.

Title:	Developing Bloom Filters for Web Archives' Holdings
Author(s):	Klein, Martin Balakireva, Lyudmila Leonidovna Hulob, Karolina Rudomino, Ingeborg Celjak, Drazenko
Intended for:	Report
Issued:	2021-12-17

Developing Bloom Filters for Web Archives' Holdings

Martin Klein, Lyudmila Balakireva
Los Alamos National Laboratory (LANL)
{@mart1nkle1n, @milbala}

Karolina Holub, Inge Rudomino
National and University Library in Zagreb, Croatian Web Archive (HAW)
{@KarolHolu, @ingeRudomino}

Drazenko Celjak
SRCE - University of Zagreb University Computing Centre
@dceljak

Thanks to the IIPC for generously funding this project!

Recall, the Problem

- Archival holdings largely unknown
  - To the public
  - To web archives

https://twitter.com/anjacks0n/status/466690812269846528

- Searching through a CDX file
  - Not available to the public
  - Requires insight knowledge and time

- CDX file sharing w/ 3rd parties
  - Not practical
  - Potential legal implications unclear

Investigating a Potential Solution

- Bloom Filters (see: https://llimllib.github.io/bloomfilter-tutorial/)
  - Data structure to reveal whether an element is present in a set
  - Database of hashed URIs that are part of a WA's holdings

- Pros:
  - Fast lookup (hash URI, check for match)
  - No active publication of plain URIs in archive
    - Benefit for dark/hybrid archives

- Unknowns:
  - Cost to create a BF
  - Performance
  - Scale

At the previous IIPC webinar…

- Sample CDX file from HAW
  - Contained 180 million URIs

- BF library:
  - Max of 200mio URLs
  - Desired FPR: 1%
  - Number of hash functions: 5
  - https://github.com/Baqend/Orestes-Bloomfilter

At the previous IIPC webinar…

Mode	**Redis DB**
Partner	**LANL**
Ingested entries	180,379,433
Time to ingest	7,273.6s (121.2min)
Size of the BF	246MB
Query response time	0.033ms
FPR	0.66%

Since then…

BF implementation in two possible modes:

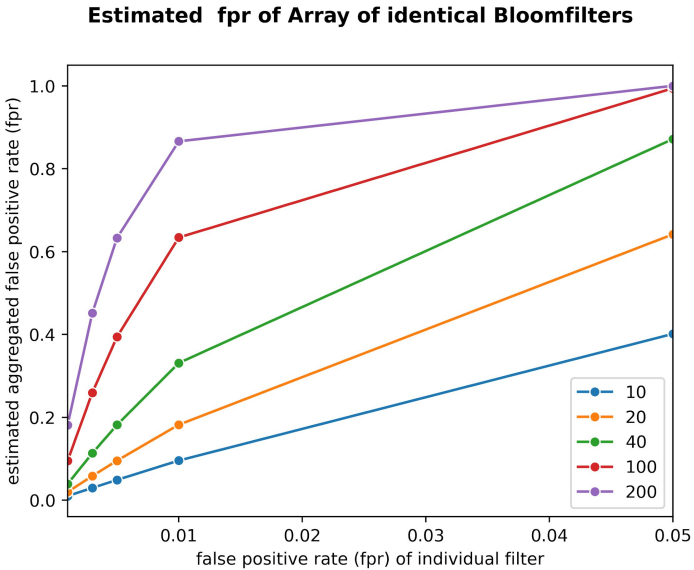
- Redis-based
  - Persistent store
  - Possibly distributed across systems
- Memory-based
  - Not persistent
  - “local”

Redis vs. Memory Mode

Mode	Redis DB		Memory	
Partner	**LANL**	**HAW**	**LANL**	**HAW**
Ingested entries	180,379,433	180,379,433	180,379,433	180,379,433
Time to ingest	7,273.6s (121.2min)	10,759.2s (179.3min)	1,382.7s (23m)	1505.5s (25.1min)
Size of the BF	246MB	246MB	246MB	246MB
Query response time	0.033ms	0.051ms	0.0008ms	0.0008ms
FPR	0.66%	0.66%	0.66%	0.66%
Time to serialize to CSV	16.8s	17.8s	2.6s	3.4s

How does this scale?

- Redis mode:
  - Key limit of 512MB, so BF can not exceed 512MB in size
  - HAW, for example, would currently require ~ 1GB
  - Dynamic BFs previously introduced
    - FPR increases significantly with each added BF
- Memory mode:
  - Got unlimited memory?
- BF library:
  - uses integer values
  - BF can not grow beyond 268.44MB



Our Proposal

Modified DBFs:

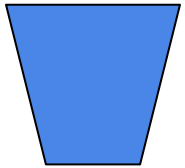
- HAW: 800mio URLs & 200mio added per year
- Creation of 16 BFs, 200mio URLs each
- Name each BF after a hex character (‘0’, ’1’, ’2’, ’3’, ’4’, ’5’, ’6’, ’7’, ’8’, ’9’, ’a’, ’b’, ’c’, ’d’, ’e’, and ’f’)
- special routing rule for insertion, based on first hex character of the URL’s MD5 hash value
- each URL will be inserted in its corresponding filter, matched by name
- Distribution done by hash values → filters have equal population of URLs (same probability for insertion)

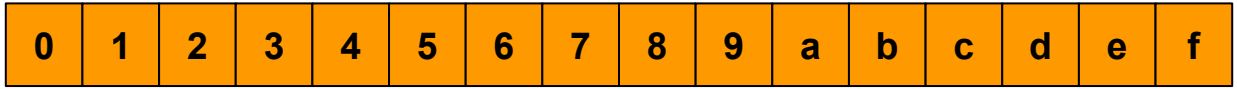
Our Proposal

Modified DBFs:

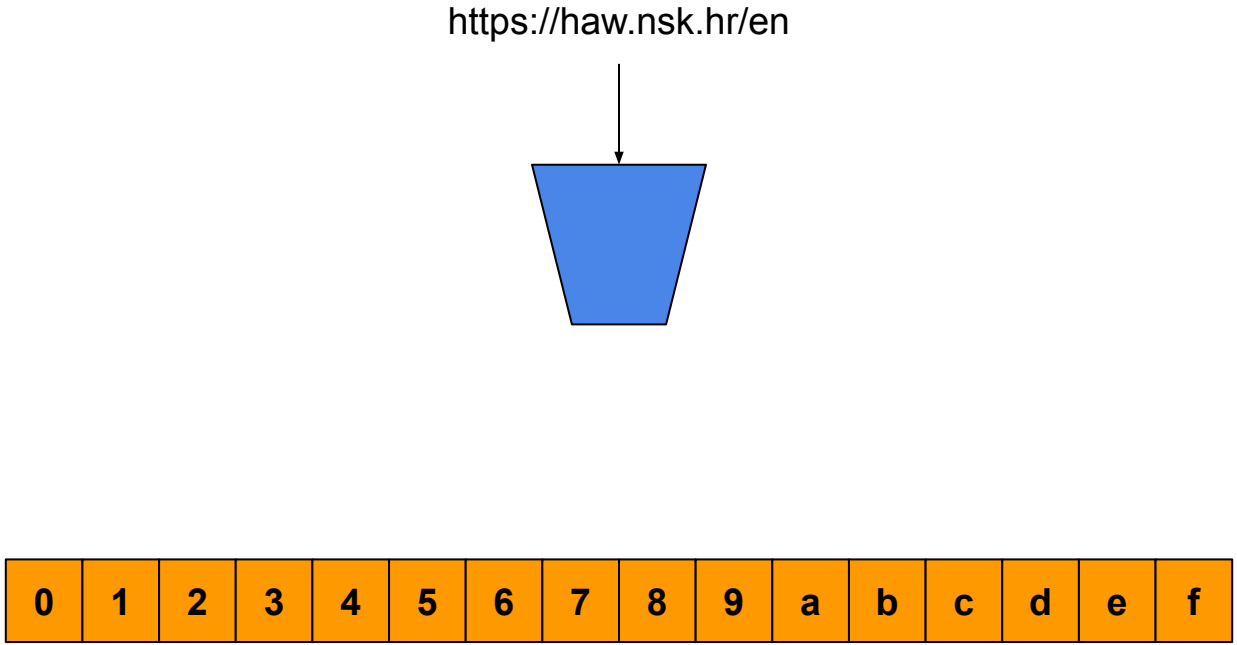
- Lookup done by routing for appropriate BF based on URL's first hex character
- Avoids parallel querying, main bottleneck in DBFs
- For HAW, 16 BFs:
  - Desired FPR: 1% at 3.8 GB
  - Desired FPR: 5% at 1.6 GB
  - Good for next 10+ years
- Can be extended to first 2 hex characters, or 32 BFs, or ….

Our Proposal - Ingest/Lookup

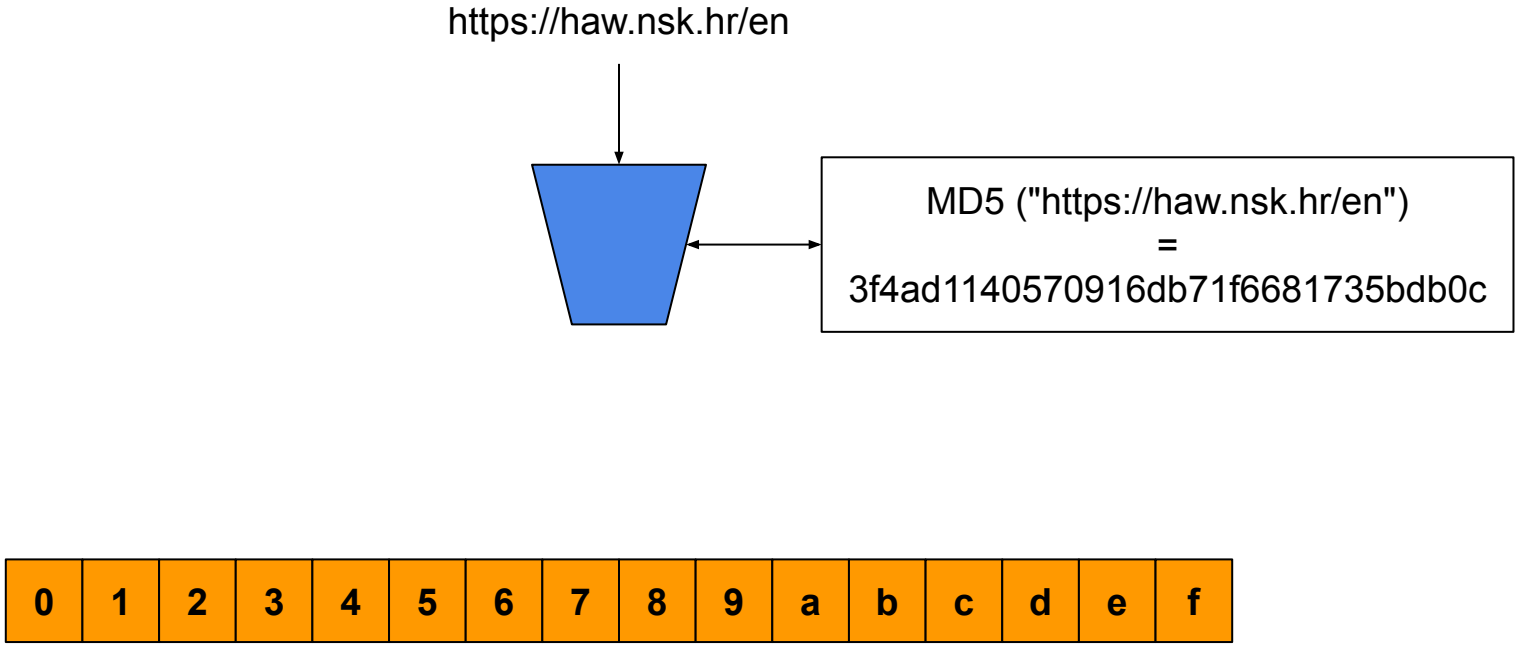




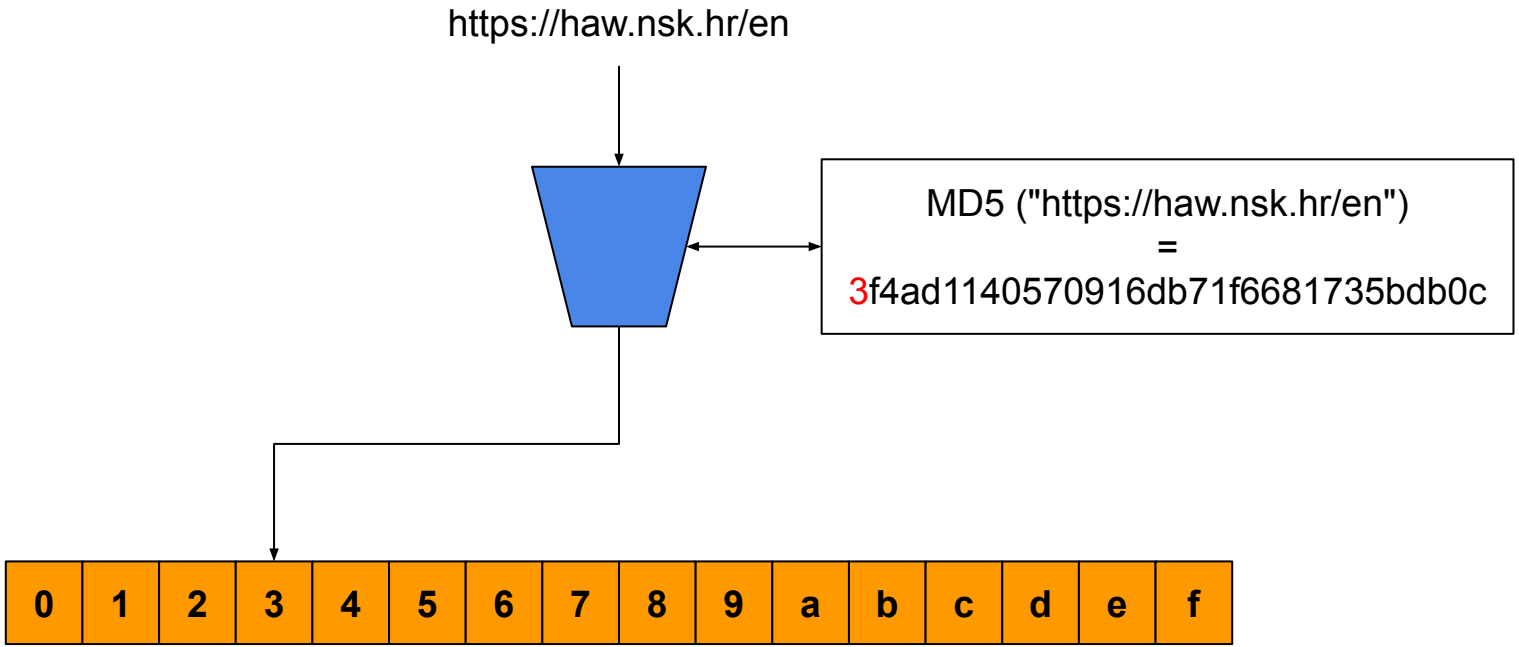
Our Proposal - Ingest/Lookup



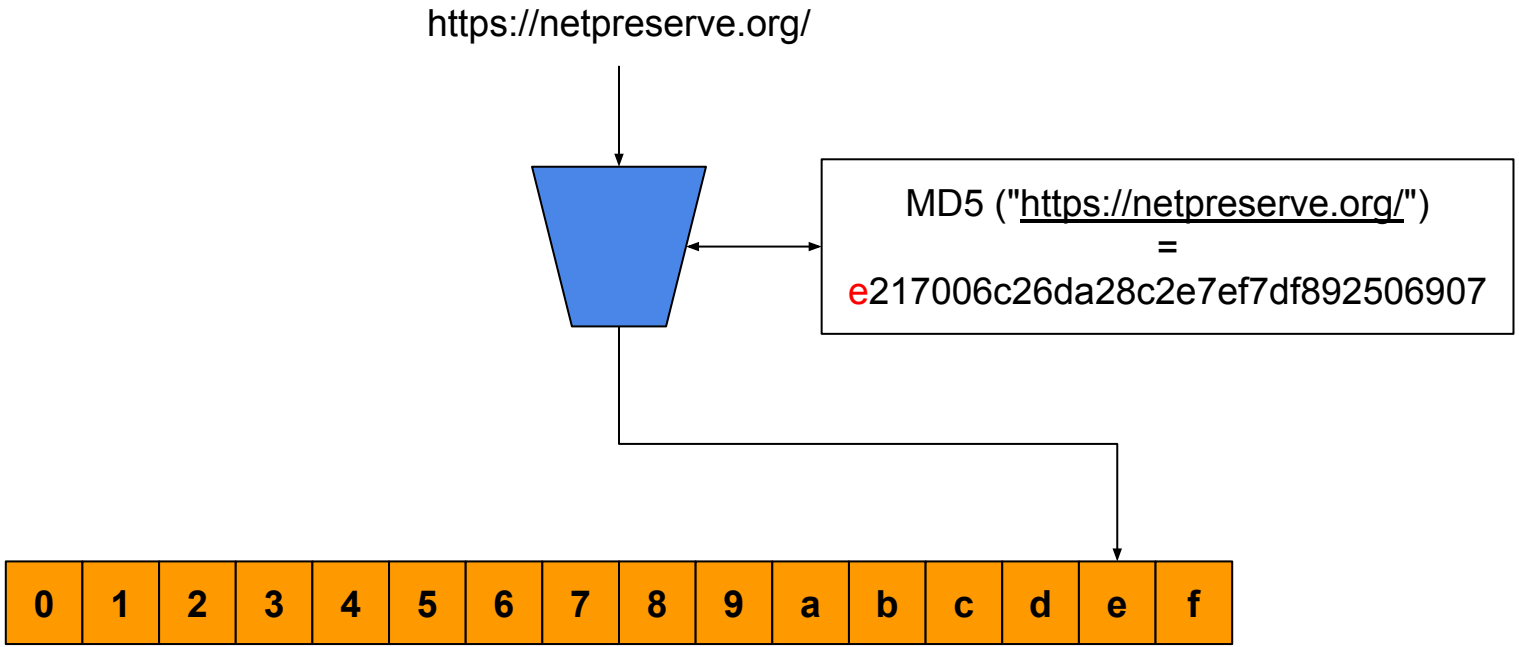
Our Proposal - Ingest/Lookup



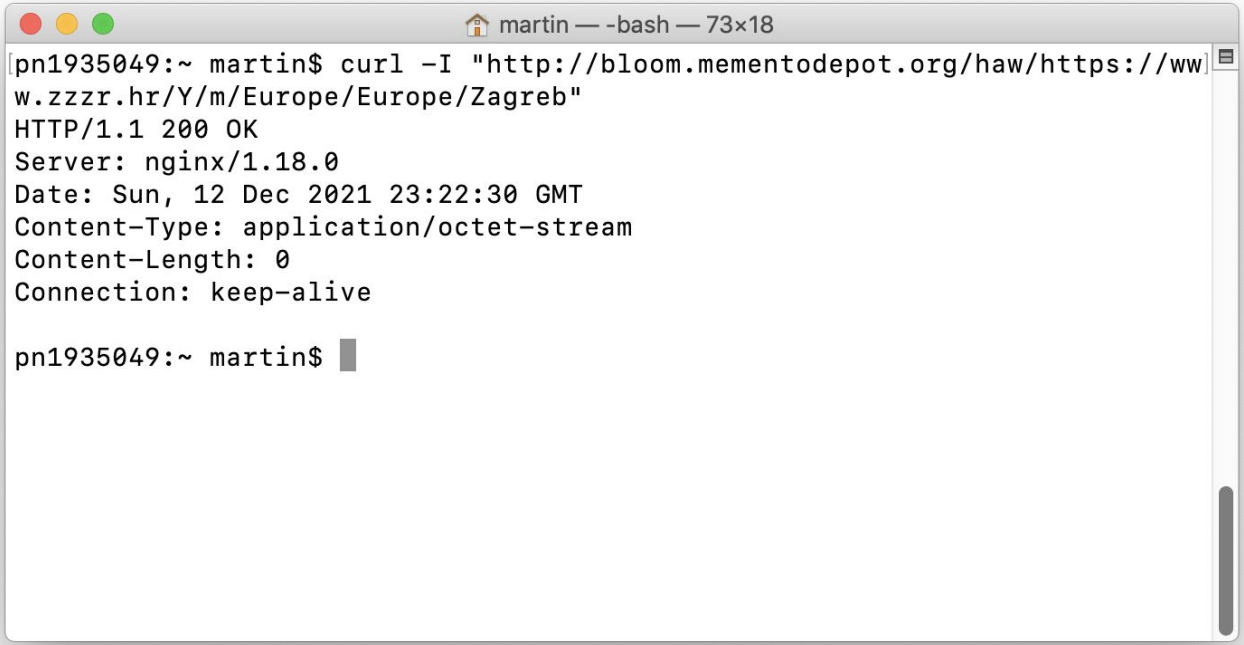
Our Proposal - Ingest/Lookup



Our Proposal - Ingest/Lookup



Our Proposal - Implemented

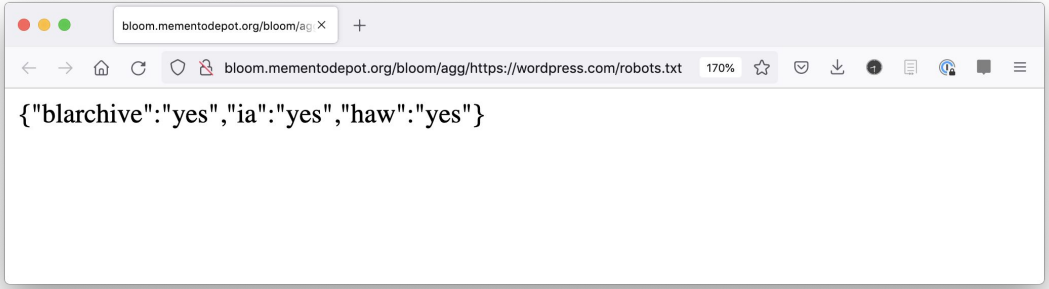


http://bloom.mementodepot.org/haw/https://www.zzzr.hr/Y/m/Europe/Europe/Zagreb

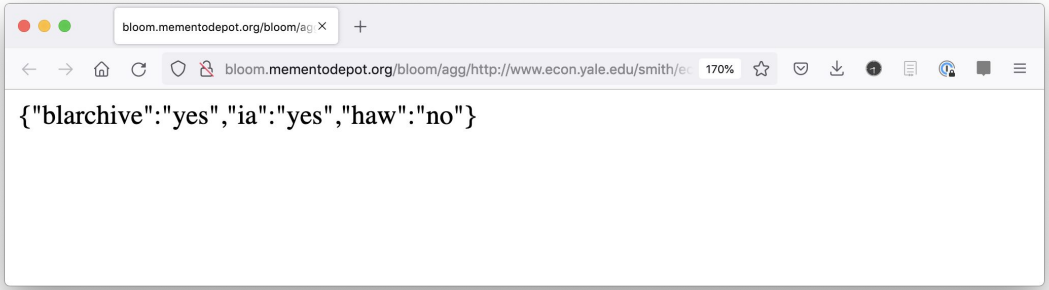
Pilot of Federated Search via BFs

- Implemented a simple lookup service, based on CDX files from:
  - HAW
  - Internet Archive
  - British Library

  - http://bloom.mementodepot.org/bloom/agg/{uri}
  - Takes URI as parameter



http://bloom.mementodepot.org/bloom/agg/https://wordpress.com/robots.txt



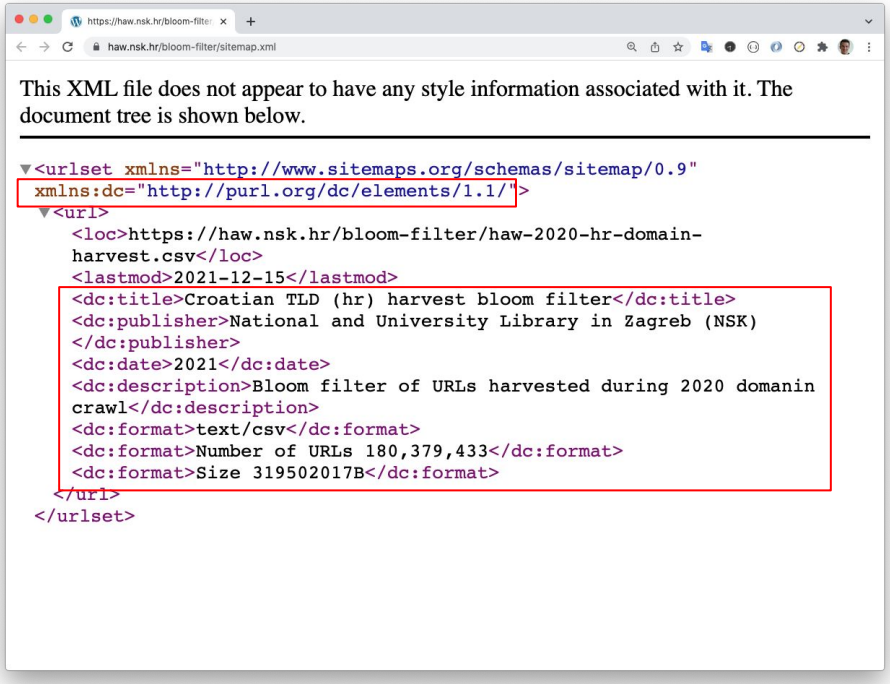
http://bloom.mementodepot.org/bloom/agg/http://www.econ.yale.edu/smith/econ116a/keynes1.pdf

Concept to Share BFs

- Croatian Web Archive (HAW)
  - Testing environment vs. production environment
  - Goal? To share archive's holdings

- The process of sharing an archive's BF has the following steps:
  - BF creation in memory
  - Serialization of BF into plain text file
  - Creation/update of sitemap, possibly with added metadata.

HAW's Sitemap



https://haw.nsk.hr/**bloom-filter**/sitemap.xml

Concept to Share (Use) BFs

- The process of using BF exposed by archives:
  - sitemap discovery
  - parse the sitemap XML file
  - extract the URL of new or updated BF files
  - get the serialized plain text file
  - ingest the file into the local BF implementation.

Final Thoughts

- Code currently undergoing LANL-internal review
- Open Source release as soon as approved

- This really works!!! (with pros and cons)

- Likely most suitable for:
  - Smaller archives
  - Individual collections
  - Live lookup of URLs during distributed crawl of (topic) collection

Developing Bloom Filters for Web Archives' Holdings

Martin Klein, Lyudmila Balakireva
Los Alamos National Laboratory (LANL)
{@mart1nkle1n, @milbala}

Karolina Holub, Inge Rudomino
National and University Library in Zagreb, Croatian Web Archive (HAW)
{@KarolHolu, @ingeRudomino}

Drazenko Celjak
SRCE - University of Zagreb
@dceljak

IIPC netpreserve.org INTERNATIONAL INTERNET PRESERVATION CONSORTIUM

Thanks to the IIPC for generously funding this project!

HAW's CDXs

File name	recordcount	loadTimeInSec	loadTimeInMin
index-2011.cdx	65,525,756	2731.744635	42.5
index-2012.cdx	69,197,158	2884.026416	48.1
index-2013.cdx	80,006,504	3339.867583	55.7
index-2014.cdx	94,445,269	3946.300632	65.8
index-2015.cdx	88,303,363	3683.1083	61.4
index-2016.cdx	93,152,112	3891.03869	64.9
index-2017.cdx	85,003,467	3801.229491	63.4
index-2018.cdx	156,308,196	6476.537223	107.9
index-2019.cdx	164,433,243	6831.800261	113.9
index-2020.cdx	180,379,433	7469.478346	124.5