A New Approach for Verifying URL Uniqueness in Web Crawlers

Wallace Henrique[1], Nivio Ziviani[1], Marco Cristo[2],

Edleno Moura[2], Altigran Silva[2], Cristiano Carvalho[1]

[1] Universidade Federal de Minas Gerais

[2] Universidade Federal do Amazonas

SPIRE, Pisa, October 19th, 2011

Where we are in Brazil

Size of the Problem

- In July 2008, Google reported 1+ trillion unique URLs in its scheduler queue
- To get to this number of unique URLs
  - Crawler starts at a set of initial pages and follows each of their links to new pages
  - Then it follows the links on those new pages to more pages, in a continuous fashion
  - In fact, they found more than 1 trillion individual URLs, as many pages are duplicates or auto-generated copies of each other

The Problem: Verifying URL Uniqueness

- Must check if
  - a URL is present in the repository and
  - the corresponding page was already collected
- Naive implementation:
  - Each verification might lead to a disk seek
- Previous work use a cache of popular URLs
- Better idea:
  - Accumulate URLs into a RAM buffer and
  - *Batch disk check* sequentially in one pass
  - Quadratic in the number of links that must pass through URL check

Related Work

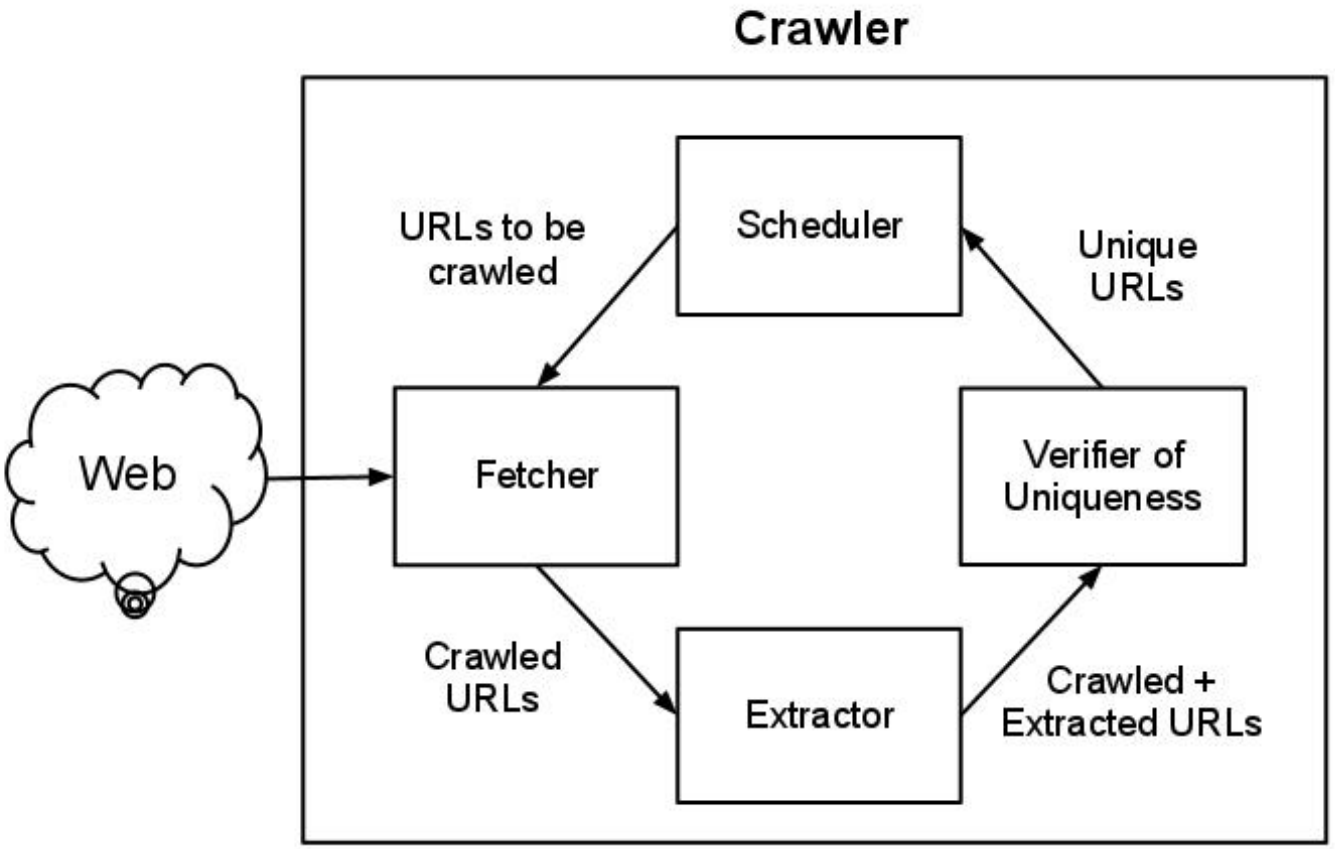
- B-tree of a DBMS
  - [Pinkerton, 1994]
- Cache of memory
  - Mercator-A [Heydon & Najork, 1999]
- Batch disk check
  - Mercator-B [Heydon & Najork 2002]
  - Polybot [Shkapenyuk & Suel 2002]
  - DRUM [Lee, Leonard, Wang & Loguinov 2009], IRLbot: Scaling to 6 Billion Pages and Beyond - baseline

VEUNIQ: VErifier of UNIQueness

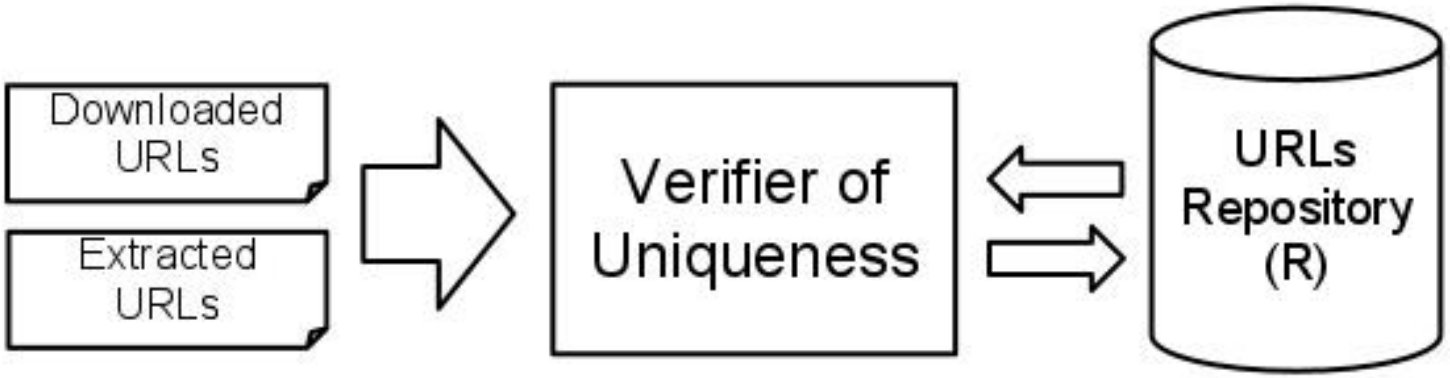
- Batch disk check strategy:
  - URLs written in lexicographical order on disk files
- VEUNIQ:
  - Organizes known URLS according to the server they belong
  - Exploits a locality of reference property:

    Distribution of links within a web page favors references to other pages from same server

Crawler Architecture: Complete Cycle

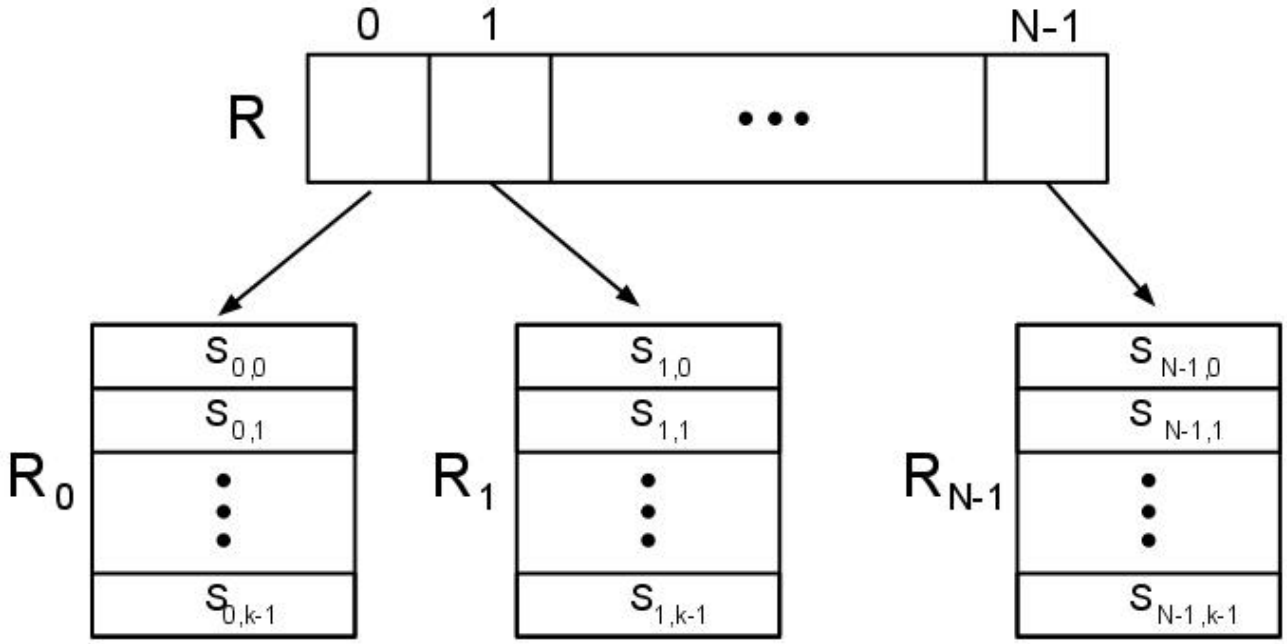


VEUNIQ I/O



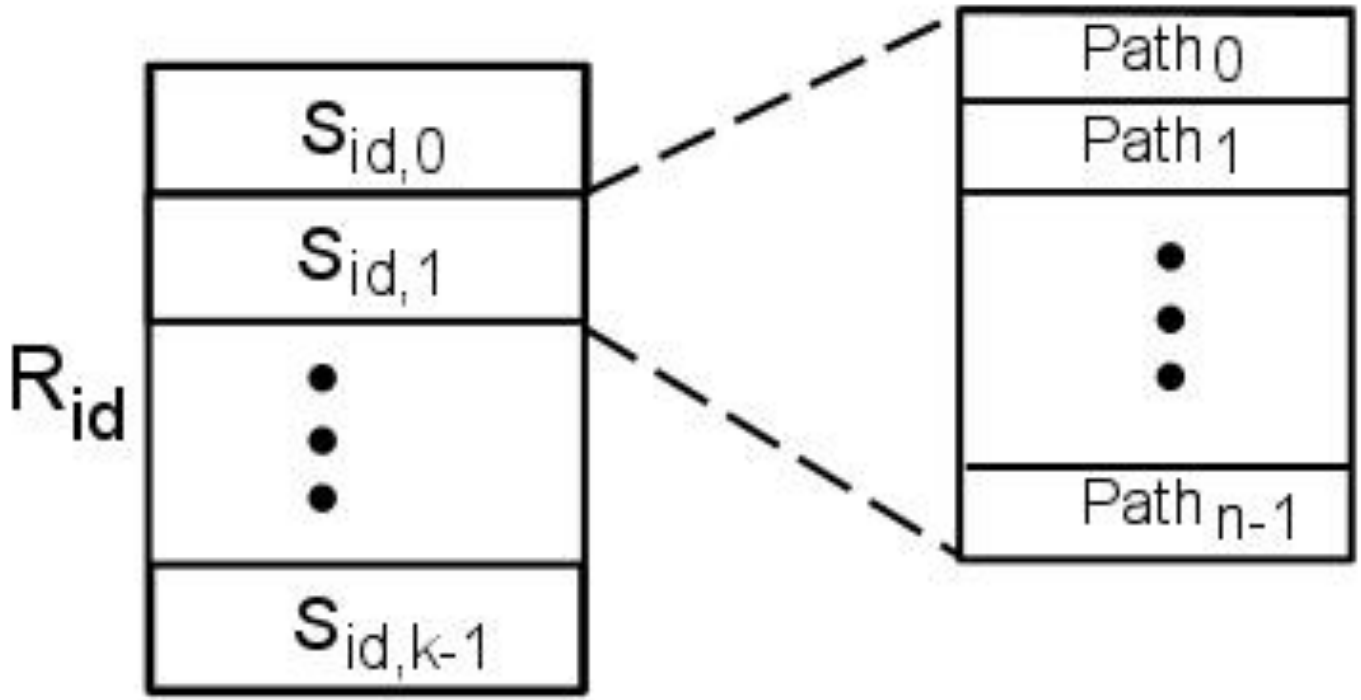
- VEUNIQ selects URLs taking into account information about the servers they belong to
- No extra cost to pre-organize the entries

Structure of Repositories R_{id} (0 <= id < N)

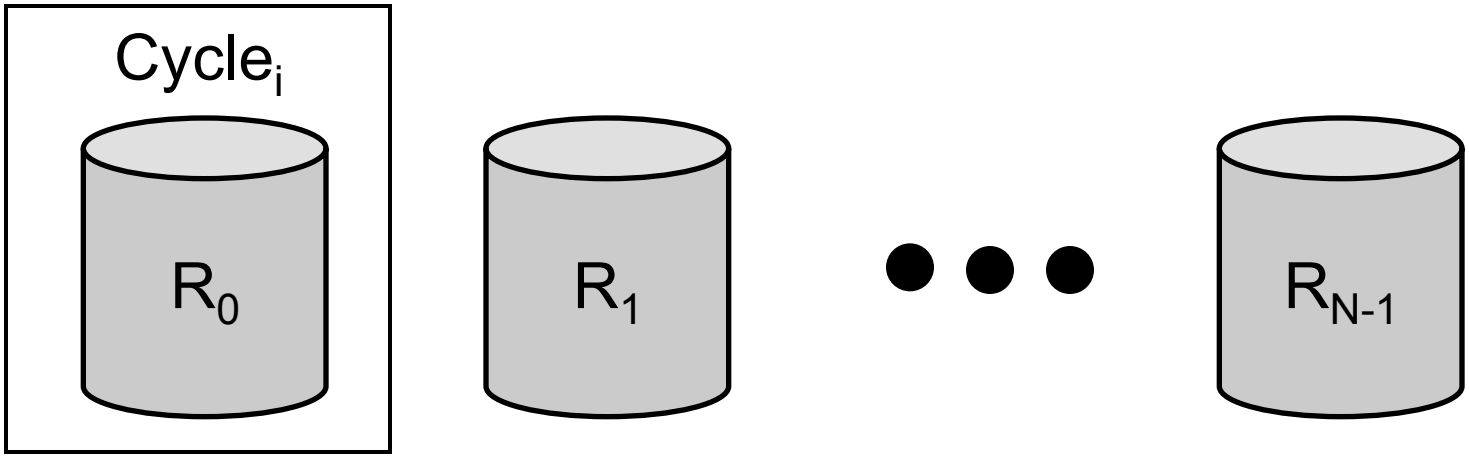


Set of unique URLs stored in N repositories R_i

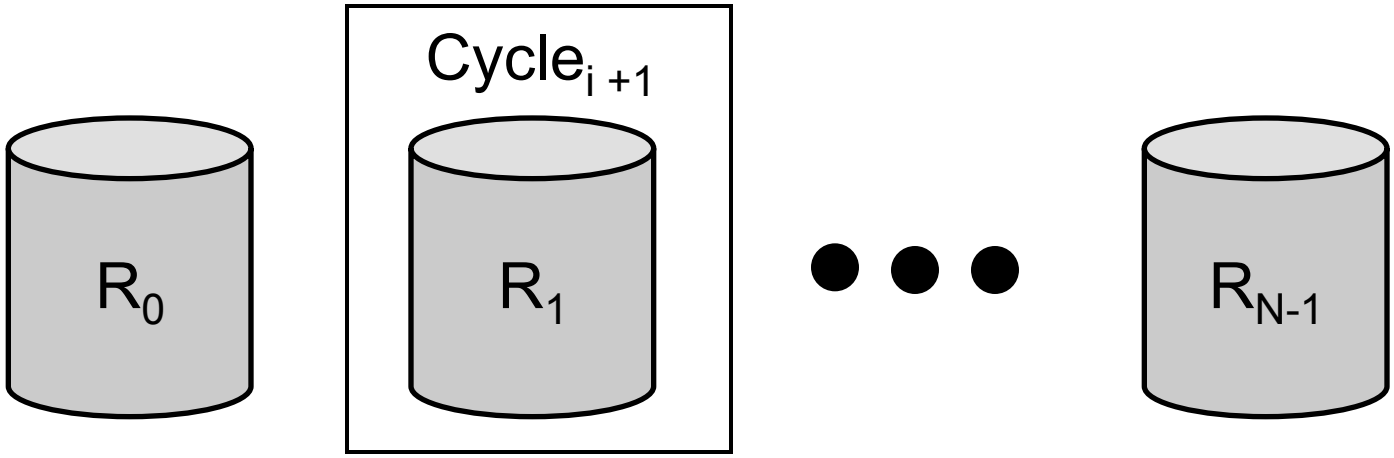
Structure of Each Repository R_{id}



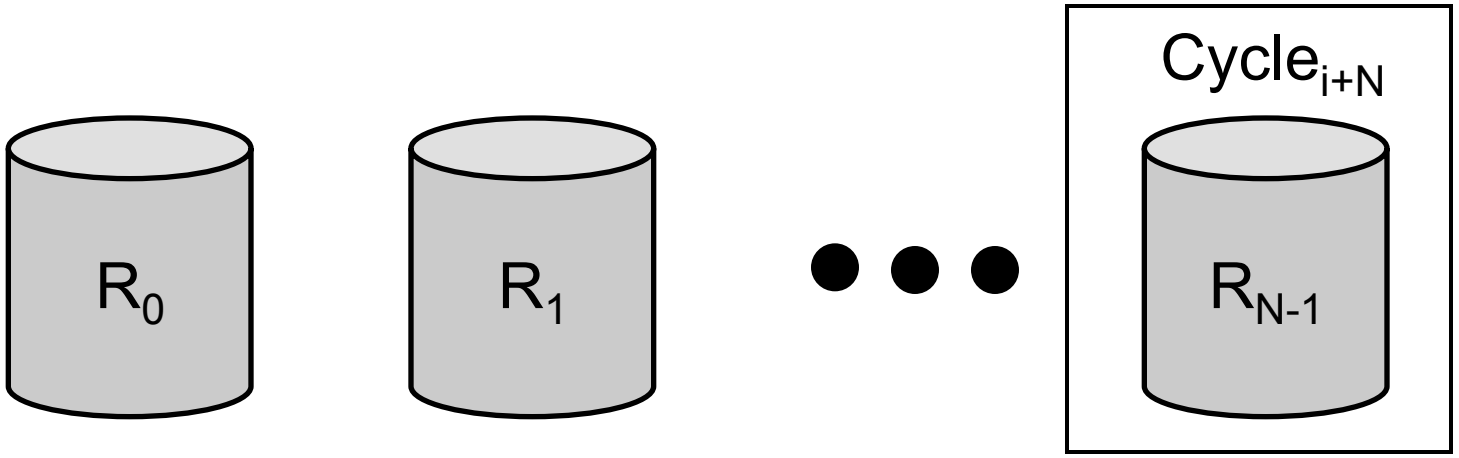
VEUNIQ: $Cycle_i$ on R_{id}, id = i mod N



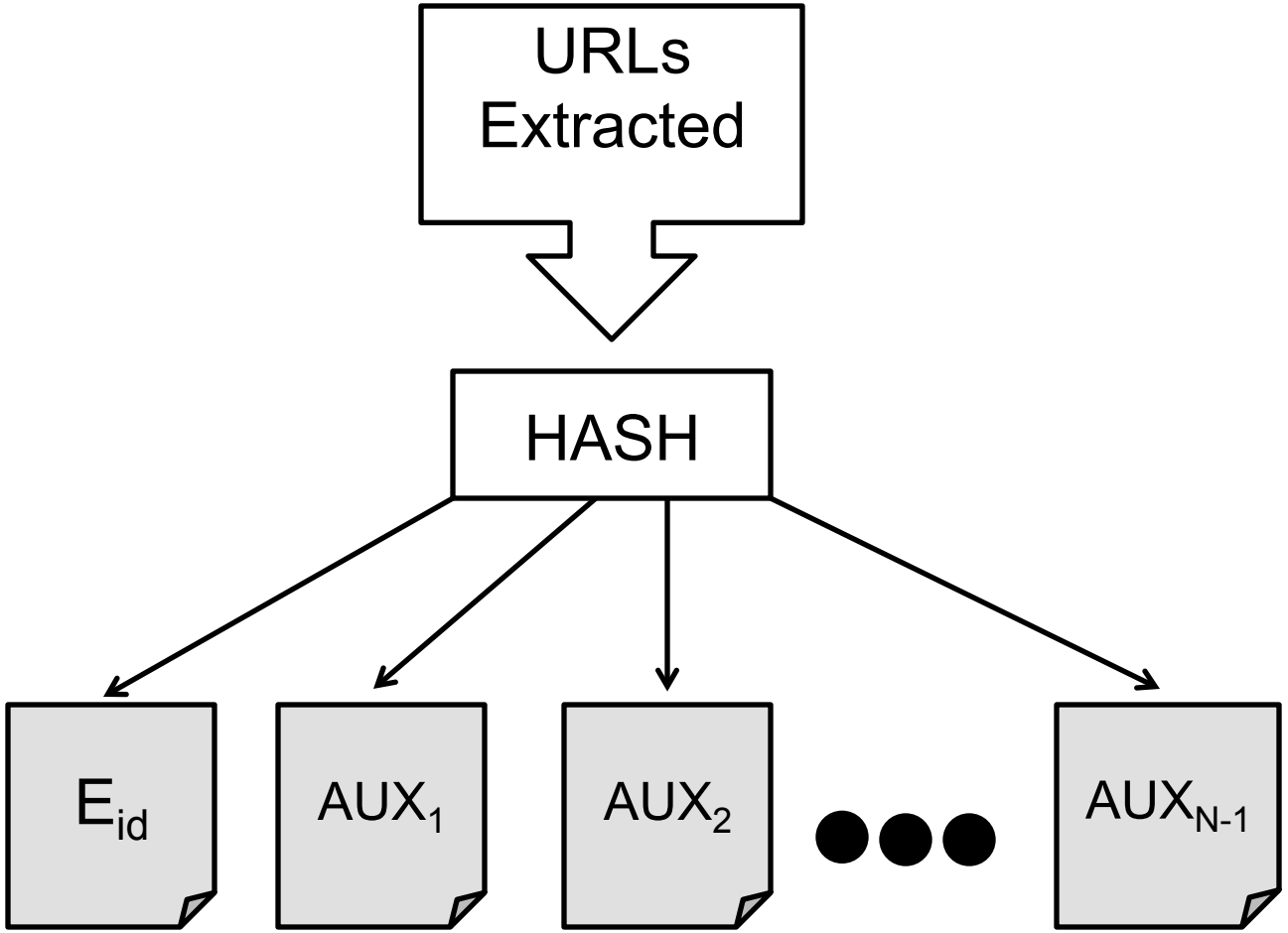
VEUNIQ: $Cycle_i$ on R_{id}, id = i mod N



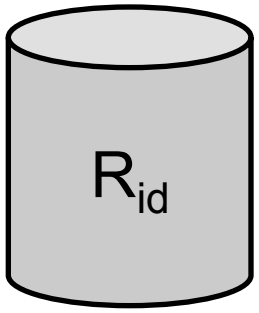
VEUNIQ: $Cycle_i$ on R_{id}, id = i mod N



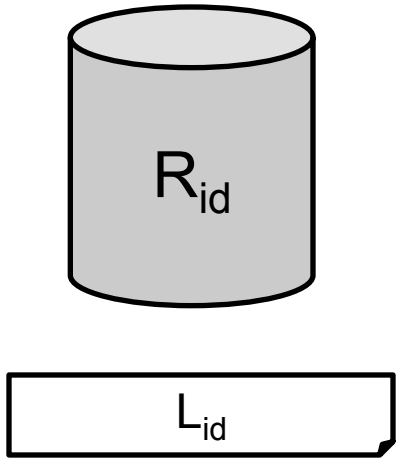
VEUNIQ: Output of URL Extractor



VEUNIQ – $Cycle_i$ on R_{id}

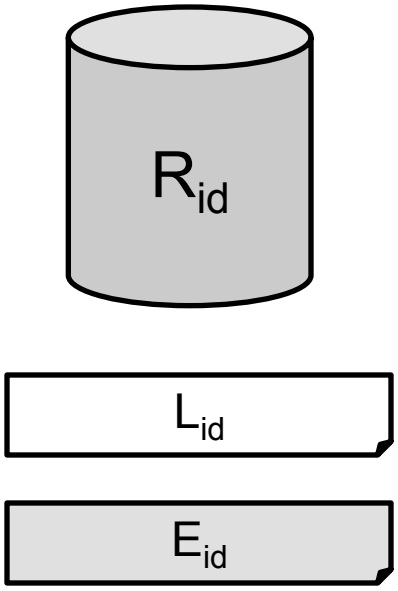


VEUNIQ – $Cycle_i$ on R_{id}



L_{id}: URLs crawled

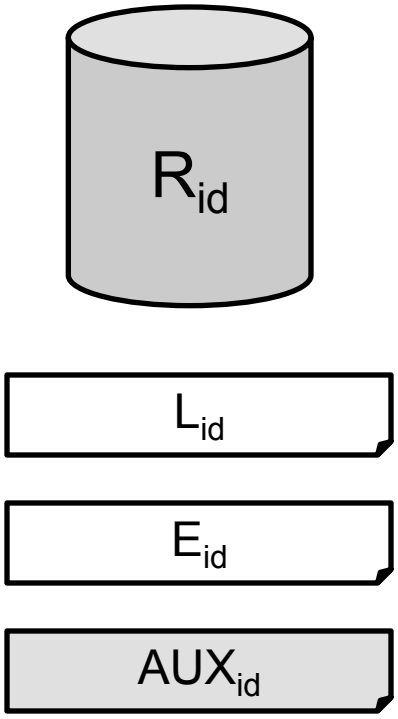
VEUNIQ – $Cycle_i$ on R_{id}



L_{id}: URLs crawled

E_{id}: URLs extracted that belong to R_{id}

VEUNIQ – One Cycle on R_{id}

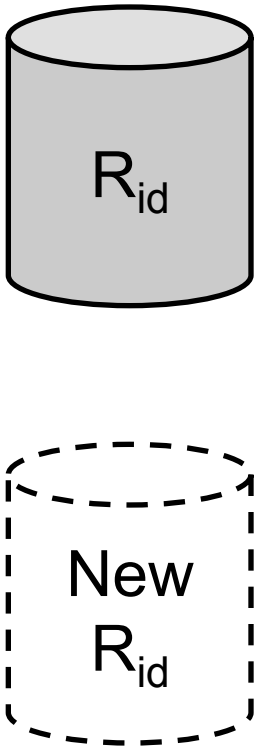


L_{id}: URLs crawled

E_{id}: URLs extracted that belong to R_{id}

AUX_{id}: URLs from previous cycles that belong to R_{id}

VEUNIQ – One Cycle on R_{id}



L_{id}: URLs crawled

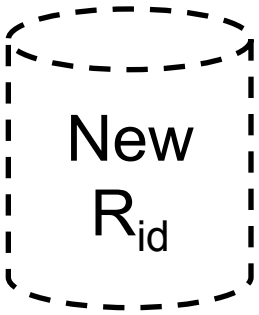
E_{id}: URLs extracted that belong to R_{id}

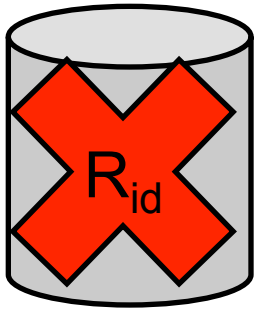
AUX_{id}: URLs from previous cycles that belong to R_{id}

New R_{id}:

Merge L_{id} + E_{id} + AUX_{id} with R_{id}

VEUNIQ – One Cycle on R_{id}





L_{id}: URLs crawled

E_{id}: URLs extracted that belong to R_{id}

AUX_{id}: URLs from previous cycles that belong to R_{id}

New R_{id}:

Merge L_{id} + E_{id} + AUX_{id} with R_{id}

DRUM (Disk Repository with Update Management)

- Keeps a set of unique URLs in a central repository R using a bucket sort strategy
- Two sets of *k* memory arrays with corresponding buckets on disk
  - For key-value pairs (KV-buckets)
  - For auxiliaries (A-buckets)
- Arrays receive KV-pairs until one reaches size r
  - All *k* buckets are merged with central repository
- Sorting is done when a KV-bucket is read
  - Efficient synchronization of all buckets with R

DRUM vs VEUNIQ

Computational Cost Model

Crawling Simulation

Computational Cost Model

n (Gb)	N	$\mathcal{R}=2$ Gb $D=22$ Tb $k=2{,}097$		$\mathcal{R}=4$ Gb $D=89$ Tb $k=4{,}194$		$\mathcal{R}=8$ Gb $D=354$ Tb $k=8{,}388$		$\mathcal{R}=16$ Gb $D=1{,}417$ Tb $k=16{,}777$	
		Drum	Veuniq	Drum	Veuniq	Drum	Veuniq	Drum	Veuniq
0.3	2,097	2.26	0.67	2.26	0.64	2.26	0.63	2.26	0.62
3	2,097	2.26	0.68	2.26	0.64	2.26	0.63	2.26	0.62
30	6,926	2.26	0.66	2.26	0.63	2.26	0.62	2.26	0.62
300	38,822	2.27	0.67	2.26	0.63	2.26	0.63	2.26	0.62
3,000	306,991	2.39	0.68	2.29	0.64	2.27	0.63	2.26	0.62

- Overhead of the two methods
  - R: RAM size, D: disk size, n: # URLs, k: # buckets
- Overhead: number of times that *bn* bytes are writen to/ read from disk (*b* is URL avg size)

DRUM vs VEUNIQ: Crawling Simulation

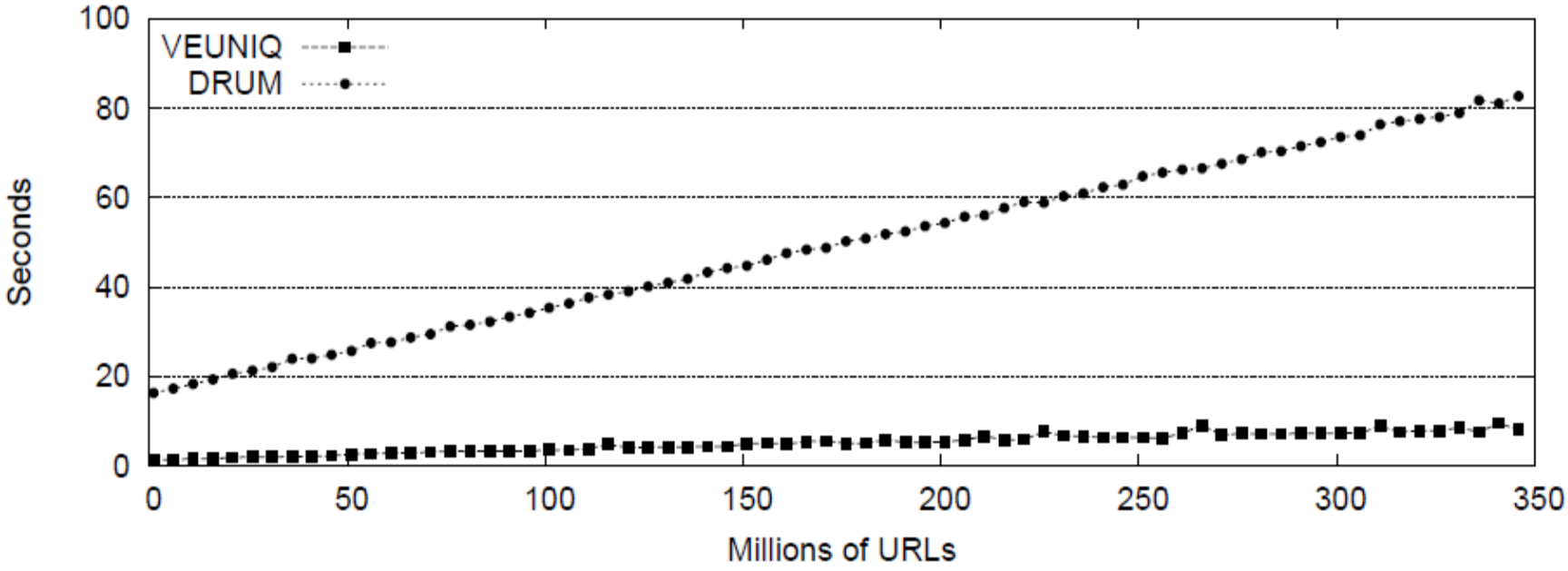
- DRUM and VEUNIQ input in each crawl cycle:
  - A set of crawled URLs
  - Metadata corresponding to these URLs
  - Set of URLs extracted from collected pages
- We used the ClueWeb09 collection
  - 1 billion Web pages occupying 25 TB
  - Crawling of 350 million URLs in 350 cycles
  - Each cycle handle 1 million URLs
    - 100,000 are collected and 900,000 extracted
  - We measured the time to update the data structures

Time to Update Repository R of URLs

Millions of URL		1	50	100	150	200	250	300	350
Time (s)	DRUM	16.28	26.39	35.62	44.70	53.99	64.46	73.18	82.85
	VEUNIQ	1.38	2.78	3.62	4.50	5.47	7.20	7.44	9.26

- We used a public C++ implementation of DRUM
- It performs final merge in a database using BerkeleyDB
  - Times reported disregard time to access database
- Set value of N to be equal to k (# of buckets in DRUM)

Time Spent in Each Crawl Cycle



- With 350 million URLs stored
  - DRUM spent 83 secs and VEUNIQ 9.3 secs

Crawling the Brazilian Web

Number of downloaded URLs :	205,084,980
Error rate: (http, timeout, empty pages, etc.)	16% (32,813,596)
Execution time:	~ 47 hours
Space:	1.3 TB
Average download:	~ 4,37 million per hour

One machine

- 64 Gb main memory
- 8 Tb external memory (4 disks of 2 TB each)

Conclusions

- VEUNIQ: accumulates URLs into a RAM buffer and check several URLs sequentially in one pass
- Same with DRUM (Disk Repository with Update Management), part of the IRLBot web crawler
- VEUNIQ: novel policy for organizing the set of unique URLs according to the server they belong to, exploiting a locality of reference property inherent in Web traversals

- Wallace Henrique, Nivio Ziviani, Marco Cristo, Edleno Moura, Altigran Silva, Cristiano Carvalho, A New Approach for Verifying URL Uniqueness in Web Crawlers, 18th Symposium on String Processing and Information Retrieval, Pisa, Italy, 2011, 237-248.

Nivio Ziviani

nivio@dcc.ufmg.br

www.dcc.ufmg.br/~nivio