Crawling like a Search Engine

A journey to crawling more than 1B pages a day

Guillaume Pitel, CTO – Sept. 2022

Babbar : a short presentation

- We make tools for SEO
  - Yourtext.guru : writing assistant for better content ranking

  - Babbar.tech : provides a wide-angle view of the web, backlinks, content, rankings => we need to crawl the WWW

Some common misconceptions

- The WWW is huge

  WRONG : the world wide web is INFINITE.

- The WWW is full of valuable information

  WRONG : the infinite majority of the WWW is crap

- Crawling the WWW is hard

  It’s certainly ugly. Is it hard ? It depends

First steps in the crawl(ing) space

A bit of history

Goal (then) : feed our Machine Learning pipeline with lots of text.

1. We used CommonCrawl's monthly dumps, which are mostly agnostic in terms of language, genre and so on.
2. Every 3 months, the CommonCrawl foundation starts a 2 weeks crawl and fetches approximately 3 Billion web pages with half of the pages never visited before.
3. But a lot of the data in common crawl is of very low quality. And we wanted to experiment on a specific subset of languages on the web

Starting our own crawling operation

- We started experimenting in 2017 and tried to crawl the good old way:
  - Get a few seed URLs, put it in a todo list
  - Fetch the urls of the todo list, analyse their links
  - Add the newly found links to the todo list
  - Repeat (with just the new ones) until we have enough pages.
- We tried first with Apache Nutch, looked at Heretrix, then stumbled upon BUbiNG[(1)] a “next generation” crawler.
- BUbiNG was amazingly fast, lightweight, easy to distribute and relatively easy to operate. Also full of bugs.

(1) Paolo Boldi, Andrea Marino, Massimo Santini, and Sebastiano Vigna. 2018

First problems

- At first, everything was fine, typical crawl speed was 600 pages/second with 8 cores.
- But as time passed, we realized we were crawling more and more crap.
- Also, BUbiNG holds the whole web frontier in memory or in disk queues, but at some point, it starts to slow down because the frontier is too big : remember the WWW is infinite.

Basically, BUbiNG is great at crawling, because it handles robots and per IP/per host queues for politeness and because it is super efficient. But managing the WWW frontier is not its strongest point.

Next Steps : our real goals

- Our real goals (now) :
  - To have a continuous crawl and recrawl of the **most interesting parts** or the WWW.
  - To compute **PageRank-like metrics** on the WWW graph
  - To **store & serve** the WWW graph
  - To **analyze the content** of the page and create and index (albeit not a search-engine one) for thematical/semantical orientation

A new architecture

Split responsibilities

- It quickly became obvious that we couldn’t improve BUbiNG in terms of managing the what’s and when’s of crawling. Instead, we dumbed it down.
- We removed all the web frontier-related bits and connected BUbiNG to a message queue (Apache Pulsar), so it could receive crawl requests, and publish the crawl results.
- And we started a new project to send crawl requests and process crawl results.

Split responsibilities

A key requirement to add intelligence to a crawler is to be able to have all the necessary data to decide to crawl or not to crawl a given URL

- In Apache Nutch, the intelligence comes from a Crawl DB which is usually stored in Hbase, a distributed key-value row-oriented DB
- Nutch is batch oriented, between batches, one must recompute a fresh PageRank (or similar), gather data about pages, hosts, domains, etc. and update the crawl DB accordingly.
- This prevents a continuous operation of the crawler, computing pagerank on a “snapshot” is the normal way to go

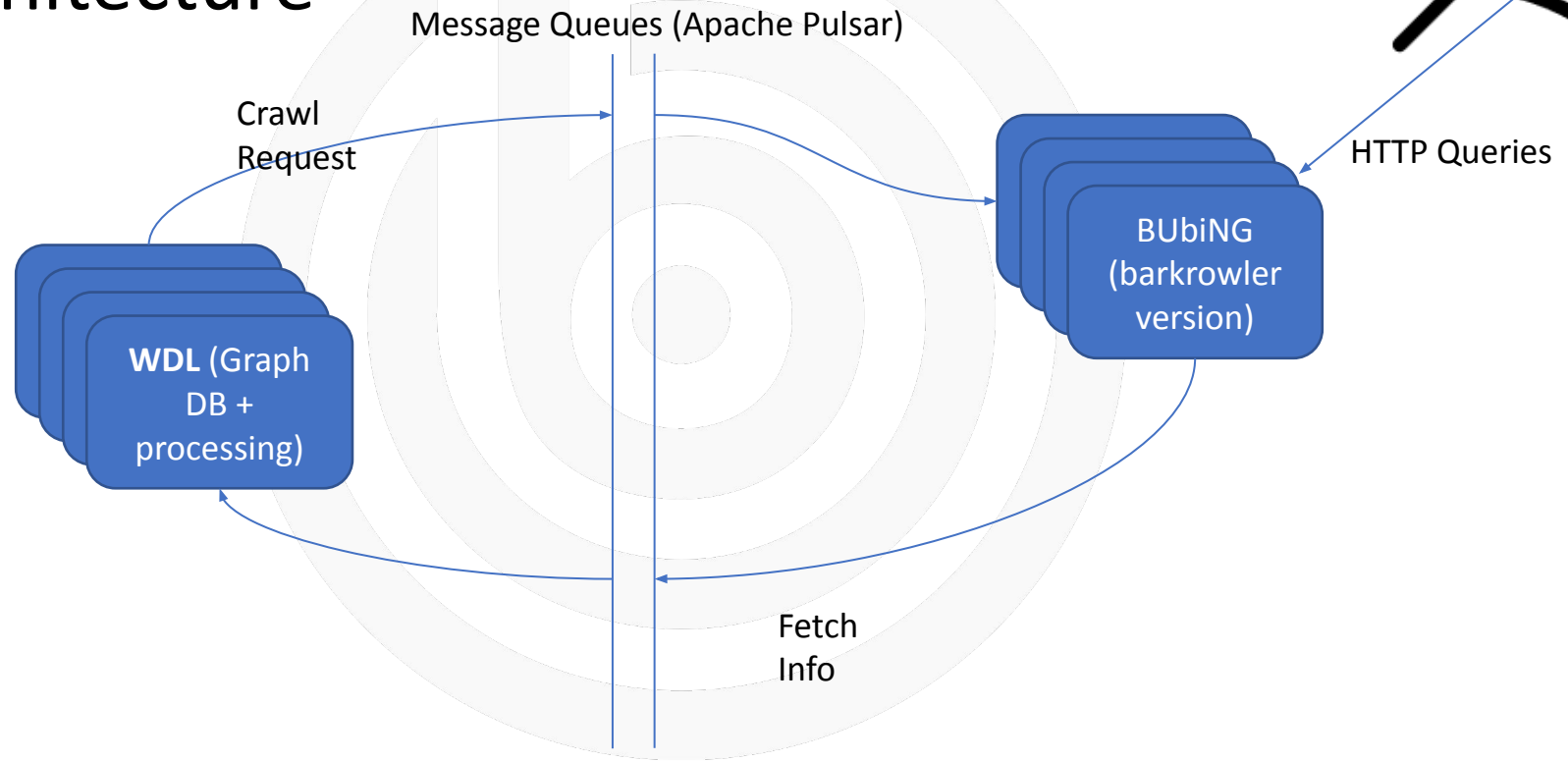
Managing the WWW graph

- We put together a distributed graph database that will also perform computation
  - A graph database normally stores nodes and links.
  - We have a bit more :
    - **Url Views** (every node of the graph, containing graph metrics)
    - **Backlinks** (because that's what really counts)
    - **Fetches** (the analysed content of a page we have crawled)
    - **Hosts** (aggregated information at host level)
    - **Domains** (same thing)
  - We also have « tables » for ip addresses and content hashes

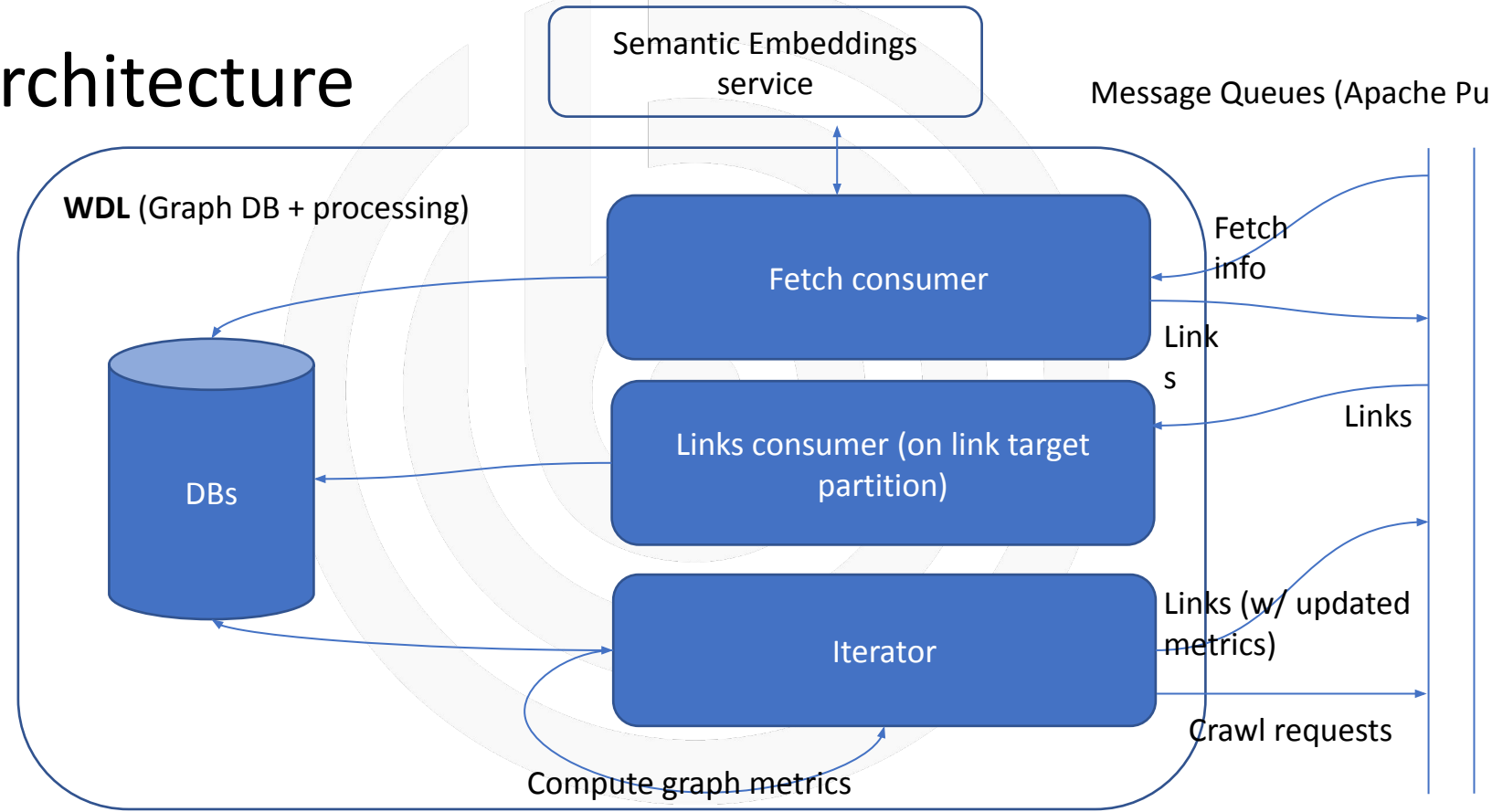
Managing the WWW graph

- We set up a dual live / delayed system
  - Live :
    - we receive information about fetched pages,
    - store it,
    - and propagate links (through the message queues because backlinks are stored with the target, not the source).
  - Delayed
    - We iterate over the databases
    - Update URLs' metrics
    - For URLs which have been fetched, progagate links with updated metrics
    - Compute aggregated metrics and statistics, update DB
    - Evaluate each URL with the crawl / recrawl policy, send crawl requests

Architecture



Architecture



Implementation details

- Compression
  - in-memory compression using homemade Huffman model for URLs
  - Semantic embeddings are highly compressed (up to 1:30 ratio)
  - DB-level compression (RocksDB)
- Rocksdb is a Log Structured Merge-Tree (LSM) DB
  - Append-only (no inplace modification)
  - Efficient merge of sorted levels (compactions)
- How to use an LSM ?
  - Avoid READ/UPDATE, instead use merges for maximum throughput
  - Lots of tuning (Tree shape, Buffer Sizes, Trigger thresholds)
  - Iterators are great, point lookup is great too but avoid for maximum performance

Crawling your website ?

Some people don’t like being crawled

We try really hard to be good citizens

- Per-IP and per-host delay between crawls
- We respect robots.txt with crawl-delay
- We support HTTP 429 (too many request) to slow down crawling, but very few web servers are configured for it

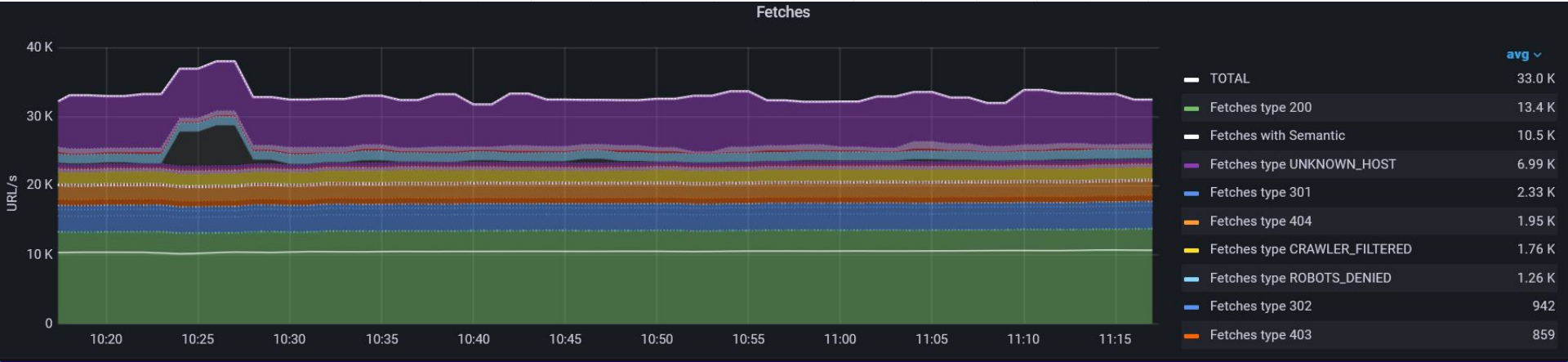
Some people don’t like being crawled

- We do receive some abuse notice, of course (not much compared to the amount we crawl). I also received two phone calls from concerned webmasters !
- Very often webmasters don’t bother reading our web page explaining the crawling, even if it’s in the User-Agent string
- Some block us right away with various methods
  - Robots.txt (ok, that’s fine)
  - Firewall (ugh… timeouts)
  - Htaccess (403, 401, 503, 200 with fake content) => bad idea, especially if on robots.txt
- Bans can be made based on IP address or User-Agent

Numbers !

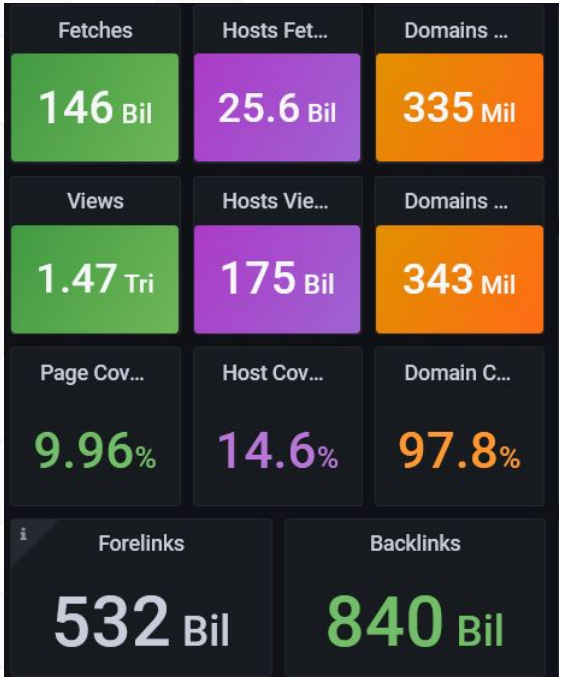
Crawling statistics

- 4 crawlers (with 2 32-cores AMD cpus), 16 IP addresses
- We crawl 20k/s HTTP responses, 32k/s crawl responses (these include non HTTP cases like network errors)
- So that's about 3B urls "crawled" per day.



The graph database

- We started crawling for real approximately 1 ½ years ago
- Since the beginning we have crawled 796B non unique urls, and 146B unique ones
- Domains are counted using ETLD (for instance .co.uk is not a TLD, but it's an ETLD)



The graph database

- We have 60 servers for storing and computing the primary WWW graph
- Total used storage size : 180TB (only !)
- Domains are counted using ETLD (for instance .co.uk is not a TLD, but it’s an ETLD)

Thank you !

Guillaume.pitel@babbar.tech

CTO Babbar