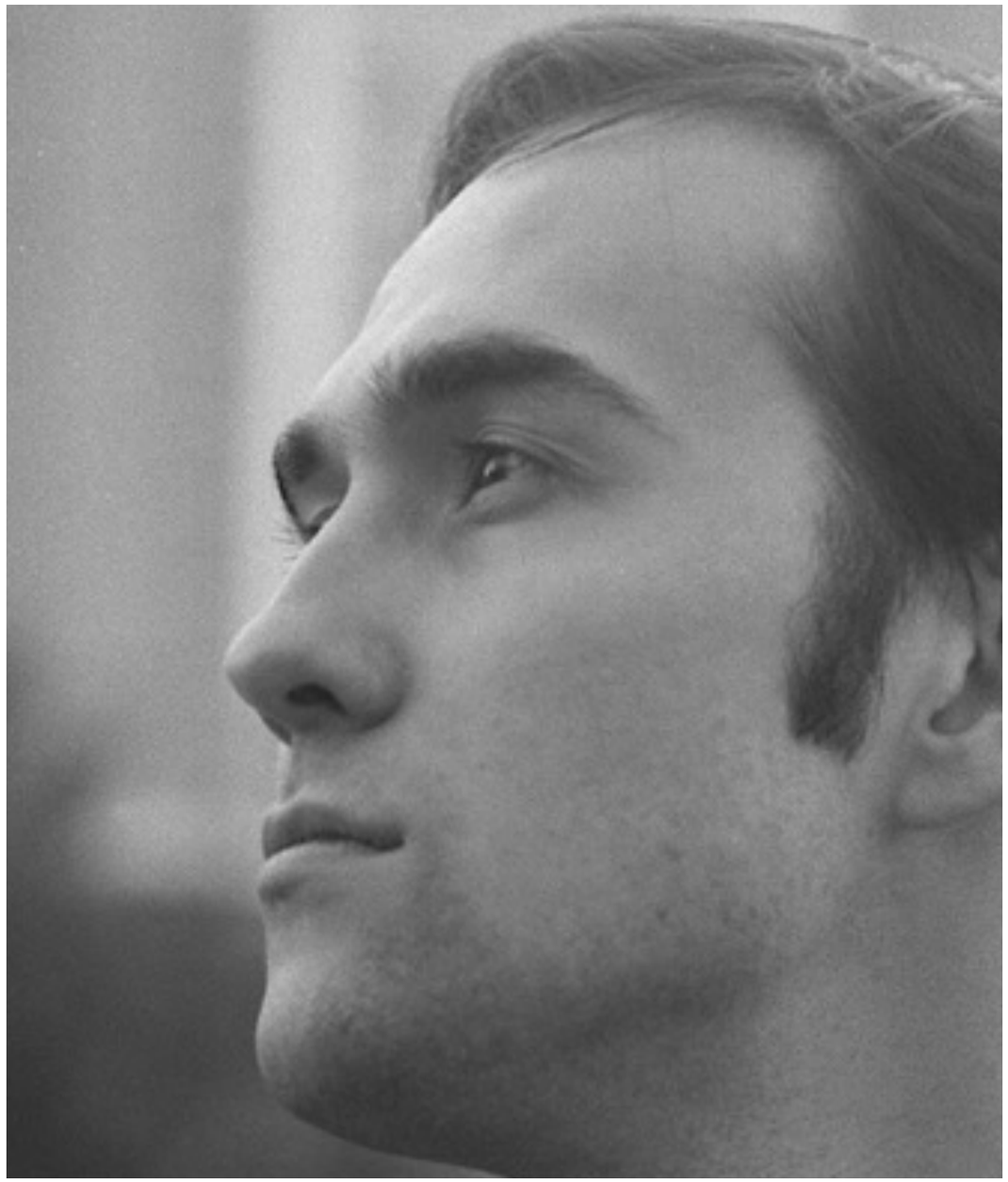
Frontera: Large-Scale Open Source Web Crawling Framework

Alexander Sibiryakov, 20 July 2015
sibiryakov@scrapinghub.com

Hola los participantes!

- Born in Yekaterinburg, RU
- 5 years at Yandex, search quality department: social and QA search, snippets.
- 2 years at Avast! antivirus, research team: automatic false positive solving, large scale prediction of malicious download attempts.



«A Web crawler starts with a list of URLs to visit, called the seeds. As the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit, called the **crawl frontier**.».

–Wikipedia: Web Crawler article, July 2015

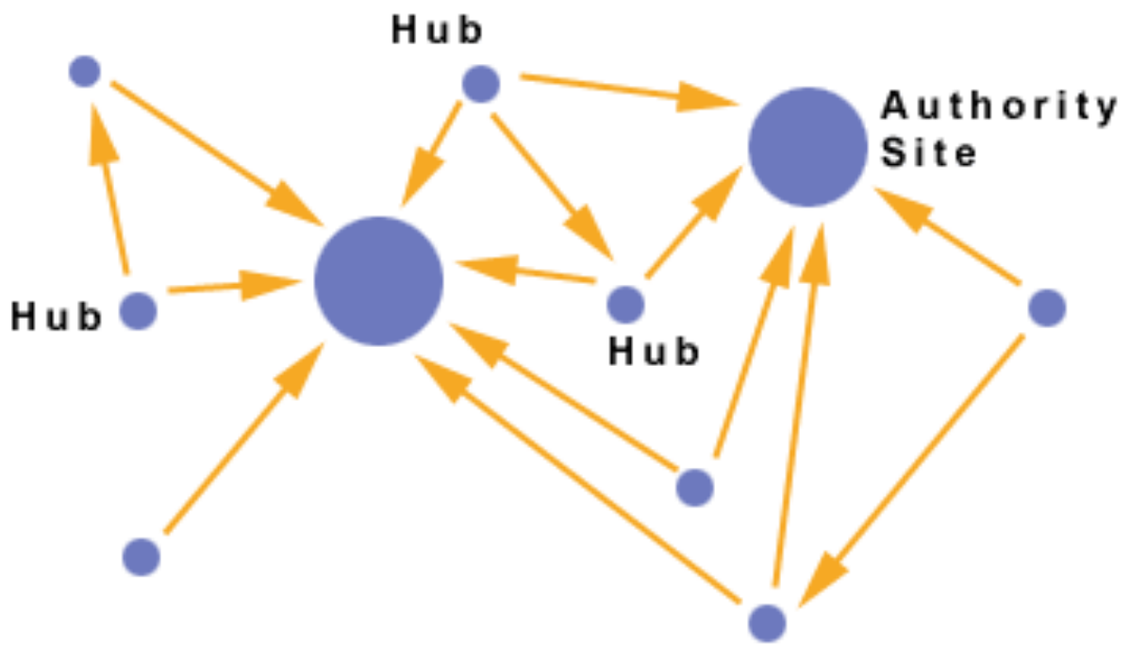

tophdart.com

Motivation

- Client needed to crawl 1B+ pages/week, and identify frequently changing HUB pages.

- Scrapy is hard for broad crawling and had no crawl frontier capabilities, out of the box,

- People were tend to favor Apache Nutch instead of Scrapy.



Hyperlink-Induced Topic Search,
Jon Kleinberg, 1999

Frontera: single-threaded and distributed

- Frontera is all about knowing **what** to crawl next and **when** to stop.
- Single-Threaded mode can be used for up to 100 websites (parallel downloading),
- for performance *broad crawls* there is a distributed mode.

Main features

- **Online operation:** scheduling of new batch, updating of DB state.

- **Storage abstraction:** write your own backend (sqlalchemy, HBase is included).

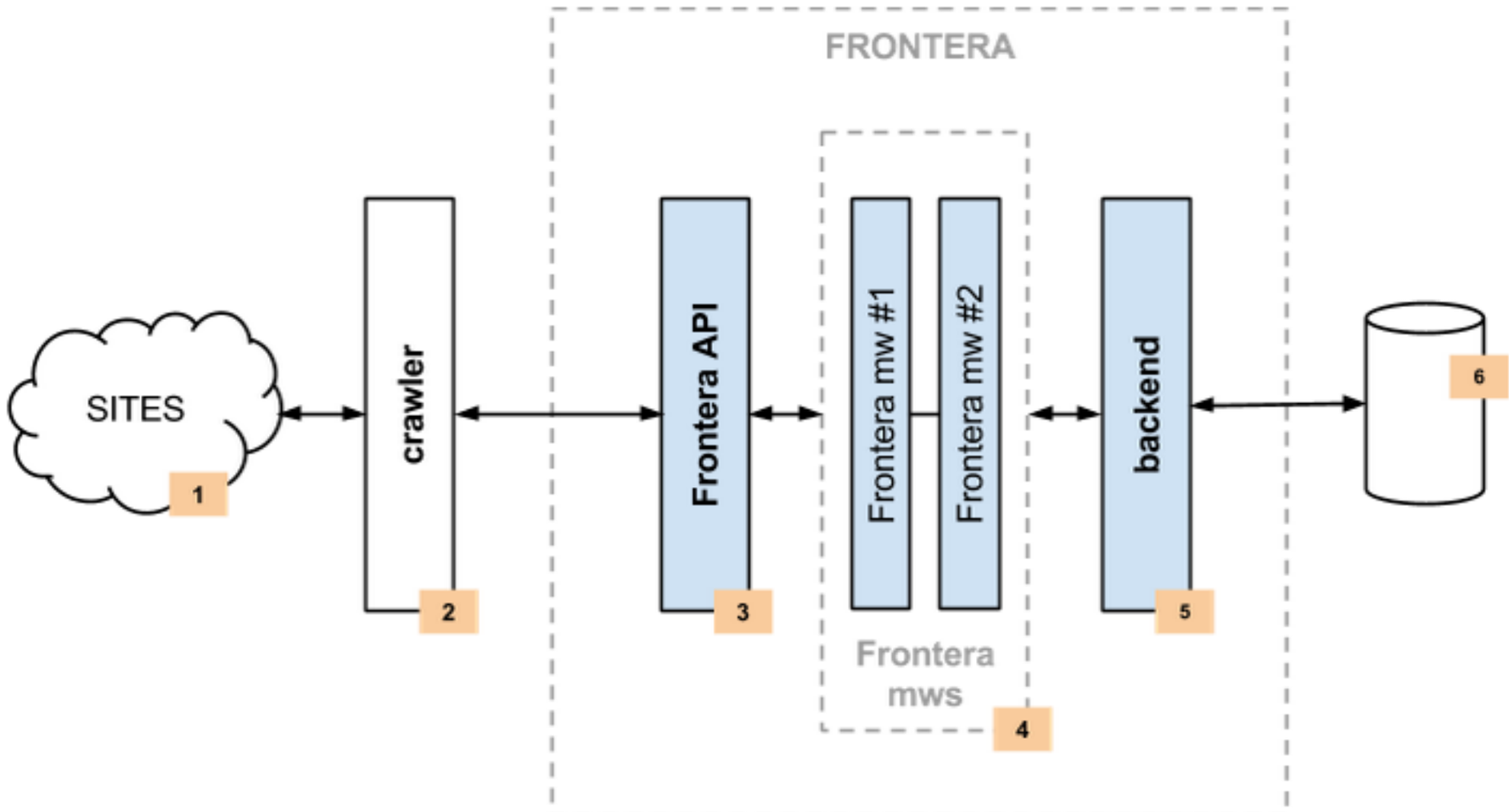
- **Canonical URLs resolution abstraction:** each document has many URLs, which to use?

- **Scrapy ecosystem:** good documentation, big community, ease of customization.

Single-threaded use cases

- Need of URL metadata and content storage,
- Need of isolation of URL ordering/queueing logic from the spider
- Advanced URL ordering logic (big websites, or revisiting)

Single-threaded architecture



Frontera and Scrapy

- Frontera is implemented as a set of custom scheduler and spider middleware for Scrapy.

- Frontera doesn't require Scrapy, and can be used separately.

- Scrapy role is process management and fetching operation.

- And we're *friends* forever!

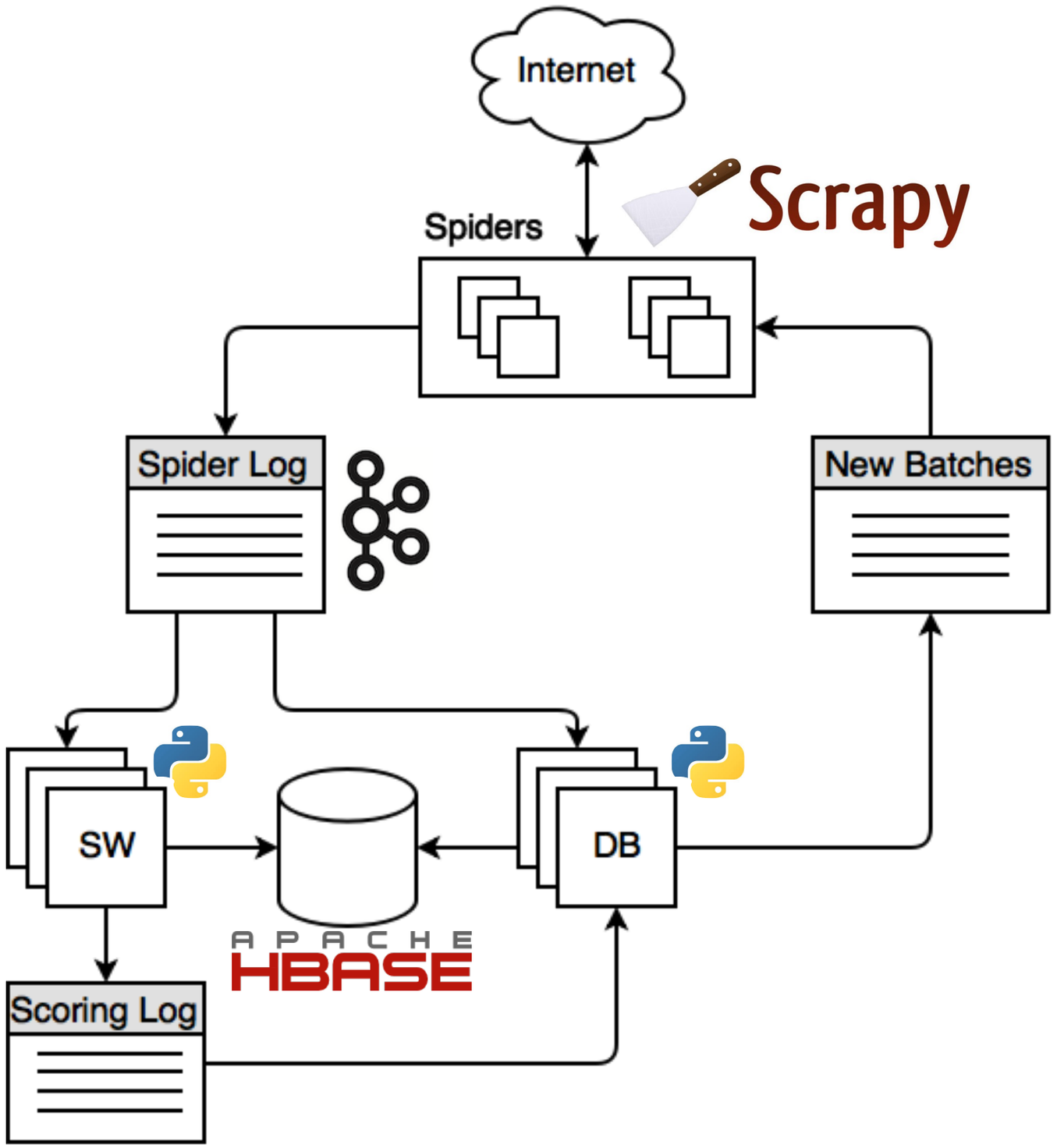
Single-threaded Frontera quickstart

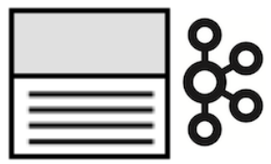
- $pip install frontera
- write a spider, or take example one from Frontera repo,
- edit spider settings.py changing scheduler and add Frontera's spider middleware,
- $scrapy crawl [your_spider]
- Check your chosen DB contents after crawl.

Distributed use cases: broad crawls

- You have set of URLs and need to revisit them (e.g. to track changes).
- Building a search engine with content retrieval from the Web.
- All kinds of research work on web graph: gathering links statistics, structure of graph, tracking domain count, etc.
- You have a topic and you want to crawl the documents about that topic.
- More general focused crawling tasks: e.g. you search for pages that are big hubs, and frequently changing in time.

Frontera architecture: distributed





Kafka topic

SW Strategy workers

DB DB workers

Main features: distributed

- Communication layer is Apache Kafka: topic partitioning, offsets mechanism.

- **Crawling strategy abstraction:** crawling goal, url ordering, scoring model is coded in separate module.

- **Polite by design**: each website is downloaded by at most one spider.

- **Python:** workers, spiders.

Software requirements

- Apache HBase,
- Apache Kafka,
- Python 2.7+,
- Scrapy 0.24+,
- DNS Service.

cloudera®

CDH (100% Open source Hadoop distribution)

Hardware requirements

- Single-thread Scrapy spider gives 1200 pages/min. from about 100 websites in parallel.
- Spiders to workers ratio is 4:1 (without content)
- 1 Gb of RAM for every SW (state cache, tunable).
- Example:
  - 12 spiders ~ 14.4K pages/min.,
  - 3 SW and 3 DB workers,
  - **Total 18 cores.**

Hardware requirements: gotchas

- Network could be a bottleneck for internal communication. Solution: increase count of network interfaces.

- HBase can be backed by HDDs, and free RAM would be great for caching the priority queue.

- Kafka throughput is key performance issue, make sure that Kafka brokers has enough IOPS.

Quickstart for distributed Frontera

- $pip install distributed-frontera
- prepare HBase and Kafka,
- simple Scrapy spider, passing links and/or content,
- configure Frontera workers and spiders,
- run workers, spiders and pull in the seeds.

  Consult http://distributed-frontera.readthedocs.org/ for more information.

Quick spanish (.es) internet crawl

- fnac.es, rakuten.es, adidas.es, equiposdefutbol2014.es, druni.es, docentesconeducacion.es - are the biggest websites
- 68.7K domains found,
- 46.5M crawled pages overall,
- 1.5 months,
- 22 websites with more than 50M pages

  For more info and graphs check the poster

Feature plans: distributed version

- Revisit strategy,
- PageRank or HITS-based strategy,
- Own url parsing and html parsing,
- Integration to Scrapinghub's paid services,
- Testing at larger scales.

Preguntas!

Thank you!

Alexander Sibiryakov,
sibiryakov@scrapinghub.com