Web Crawling

- Najork and Heydon, *High-Performance Web Crawling*, Compaq SRC Research Report 173, 2001. Also in Handbook of Massive Data Sets, Kluwer, 2001.
- Heydon and Najork, *Mercator: A scalable, extensible Web crawler*. World Wide Web , 4, 1999.
- Najork and Wiener, *Breadth-first search crawling yields high-quality pages*. Proc. 10th Int. WWW Conf., 2001.
- Arasu et al: *Searching the Web*. ACM Trans. Internet Technology, 1, 2001.

Web Crawling

$\boxed{\text{Web Crawling = Graph Traversal}}$

S = {startpage}
repeat
    remove an element s from S
    foreach (s, v)
        if v not crawled before
            insert v in S

Issues

Theoretical:

- Startset S
- Choice of s (crawl strategy)
- Refreshing of changing pages.

Practical:

- Load balancing (own resources and resources of crawled sites)
- Size of data (compact representations)
- Performance (I/Os).

Crawl Strategy

- Breath First Search
- Depth First Search
- Random
- Priority Search

Possible priorities:

- Often changing pages (how to estimate change rate?).
- Using global ranking scheme for queries (e.g. PageRank).
- Using query dependent ranking scheme for queries ("focused crawling", "collection building").

BFS is Good

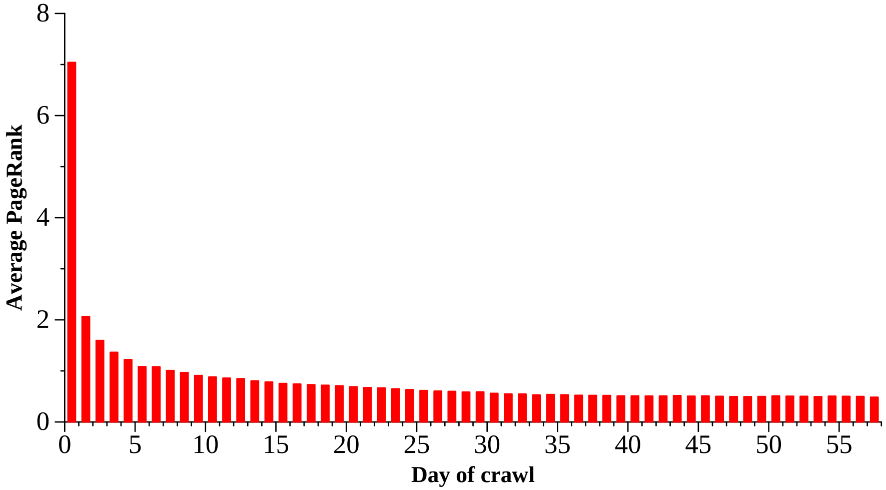


Figure 1: Average PageRank score by day of crawl

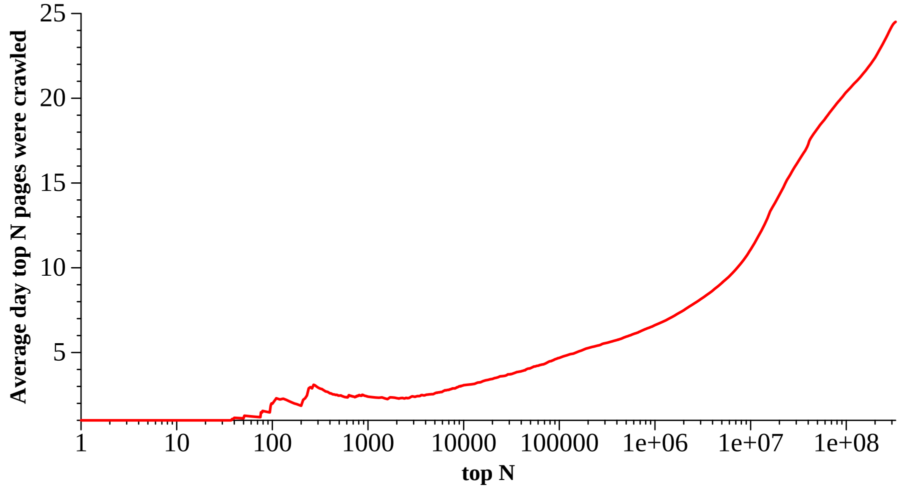


Figure 2: Average day on which the top N pages were crawled

[From: Najork and Wiener, 2001]

Statistics for crawl of 328 million pages.

PageRank Priority is Even Better

(but computationally expensive to use. . .)

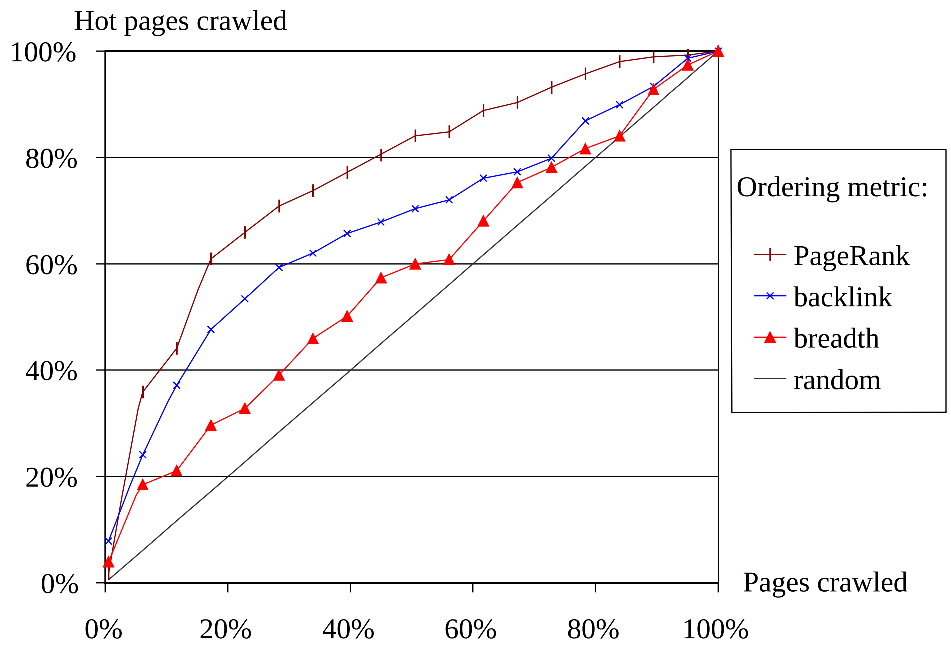


Figure 2: The performance of various ordering metrics for $IB(P)$; $G = 100$

[From: Arasu et al., 2001]

Statistics for crawl of 225.000 pages at Stanford.

Load Balancing

Own resources:

- Bandwidth (control global rate of requests)
- Storage (compact representations, compression)
- Industrial-strength crawlers must be distributed (e.g. partition the url-space)

Load Balancing

Own resources:

- Bandwidth (control global rate of requests)
- Storage (compact representations, compression)
- Industrial-strength crawlers must be distributed (e.g. partition the url-space)

Resources of others:

- BANDWIDTH. Control local rate of requests (e.g. 30 sec. between request to same site).
- Identify yourself in request. Give contact info.
- Monitor the crawl.
- Obey the Robots Exclusion Protocol (`www.robotstxt.org`). [Also read the other material there.]

Efficiency

- RAM: never enough for serious crawls. Efficient use of disk based storage important. I/O when accessing data structures is often a bottleneck.
- CPU cycles: not a problem (Java and scripting languages are fine).
- DNS lookup can be a bottleneck (as normally synchronized). Asynchronous DNS: check GNU `adns` library.

Rates reported for serious crawlers: 200-400 pages/sec.

Example: Mercator

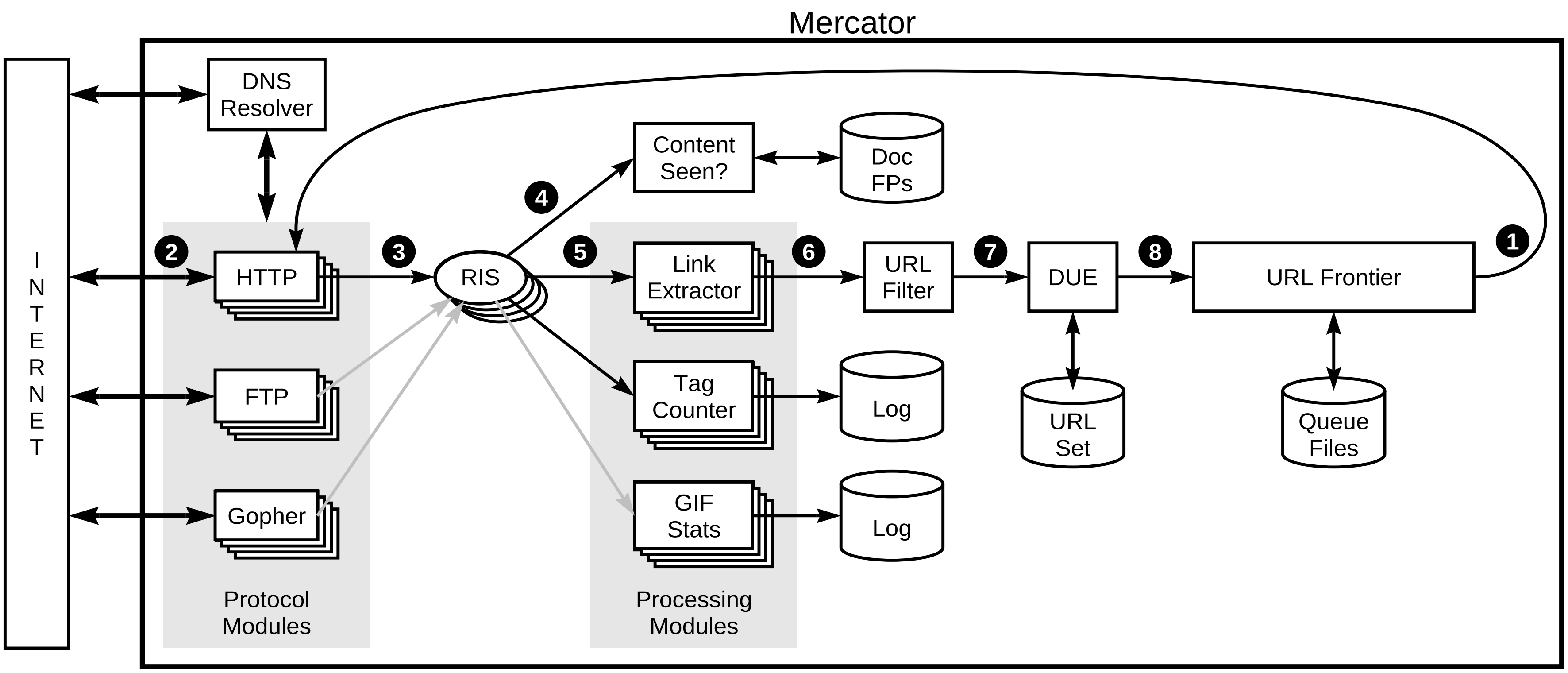


Figure 1: Mercator's main components.

[From: Najork and Heydon, 2001]

Mercator

Further ideas:

- Fingerprinting ((sparse) hashfunction on strings).
- Continuous crawling—crawled pages put back in queue (prioritized using update history).
- Checkpointing (crash recovery).
- Very modular structure.

Details: Politeness

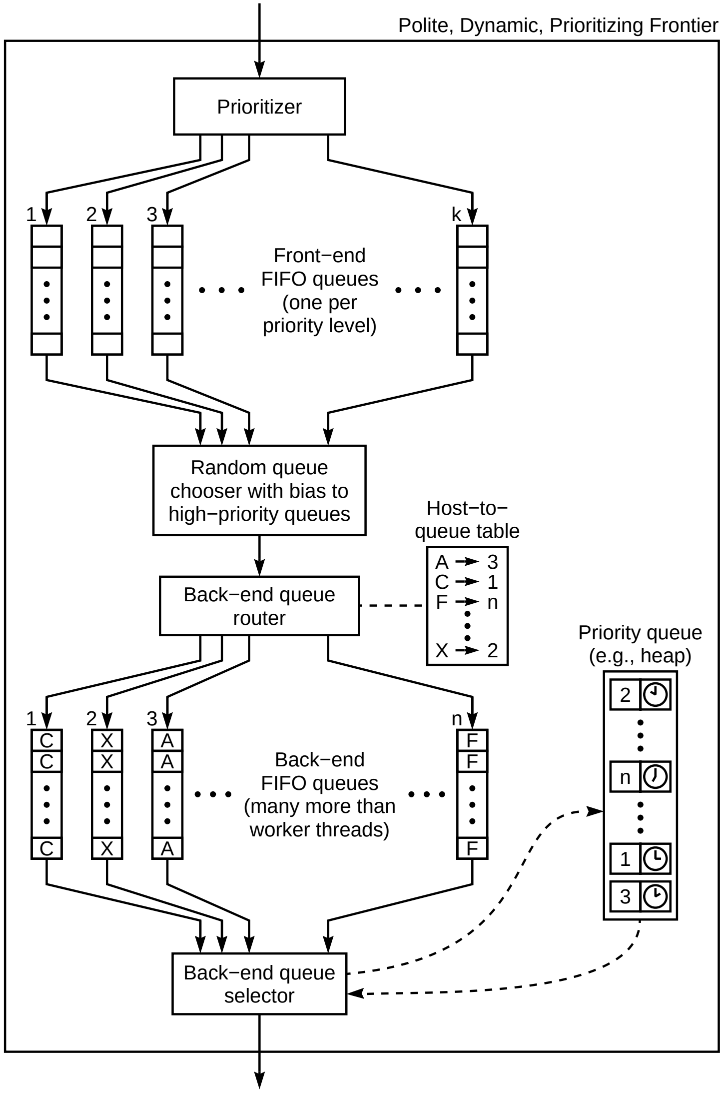


Figure 3: Our best URL frontier implementation

[From: Najork and Heydon, 2001]

Details: Efficient URL Elimination

- Fingerprinting
- Sorted file of fingerprints of seen URLs.
- Cache most used URLs.
- Non-cached URLs checked in batches (merge with file I/O).

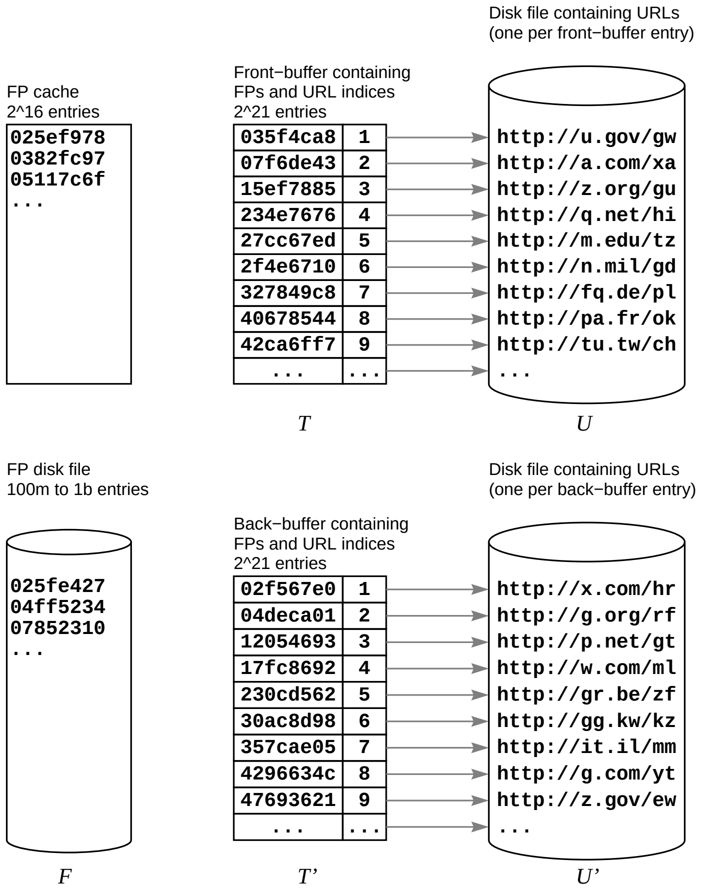


Figure 4: Our most effi cient disk-based DUE implementation

[From: Najork and Heydon, 2001]

Some Experiences

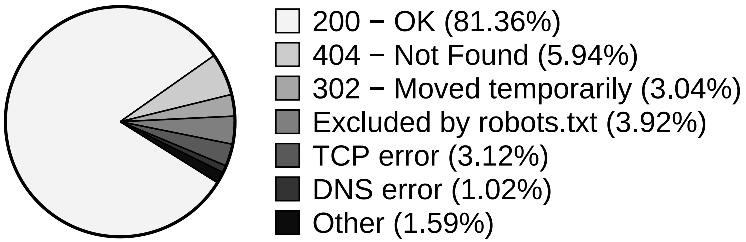


Figure 6: Outcome of download attempts

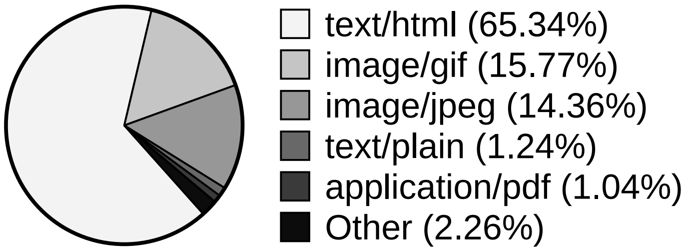


Figure 7: Distribution of content types

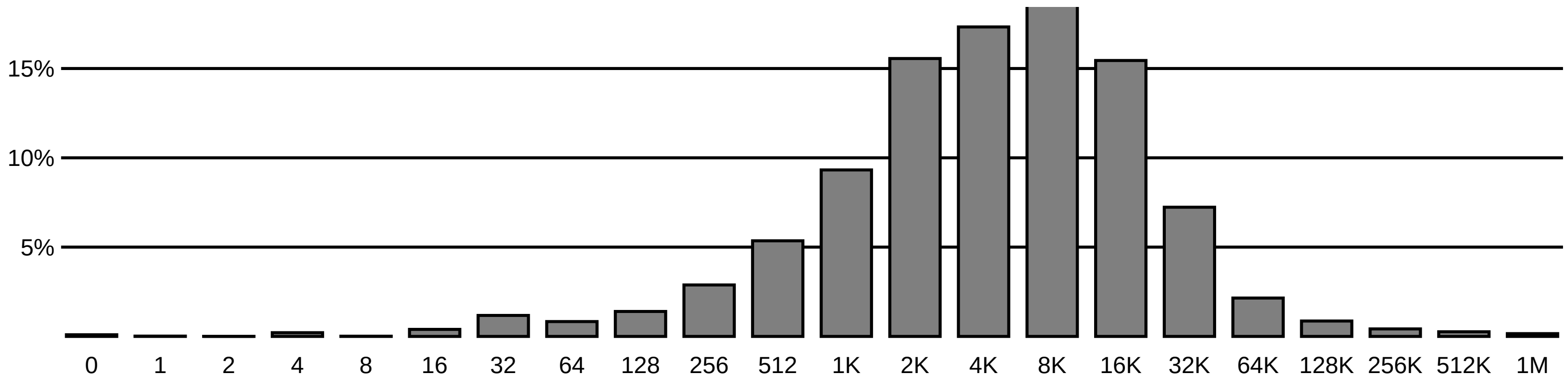
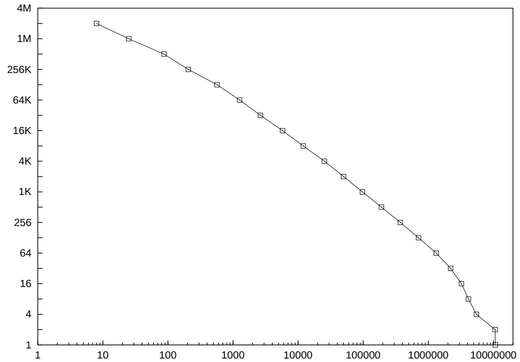
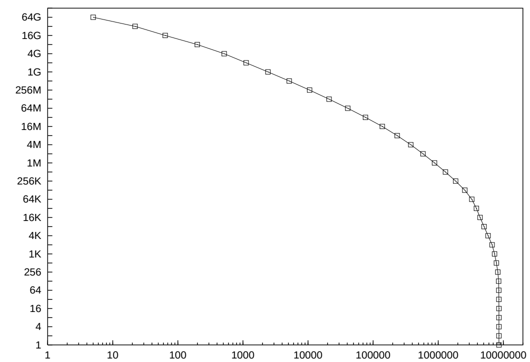


Figure 8: Distribution of document sizes

Some Experiences

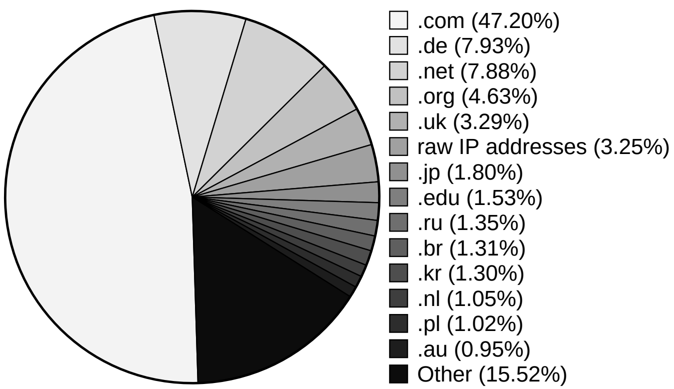


(a) Distribution of pages over web servers

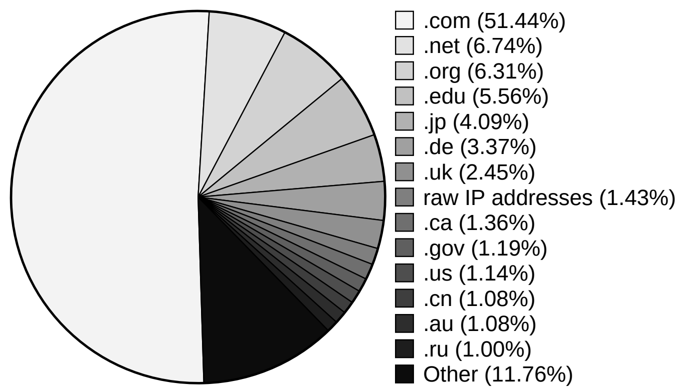


(b) Distribution of bytes over web servers

Figure 9: Document and web server size distributions



(a) Distribution of hosts over



(b) Distribution of pages over

[From: Najork and Heydon, 2001]

Further Resources

Further resources for implementing a crawler:

- Another good paper with practical info:
  Shkapenyuk and Suel: *Design and Implementation of a High-Performance Distributed Web Crawler*. IEEE Int. Conf. on Data Engineering (ICDE), February 2002.
  (`http://cis.poly.edu/suel/papers/crawl.ps`)
- HTML specification (`www.w3.org`)
- A free book on programming web agents.
  (`http://www.oreilly.com/openbook/webclient`)
- Software libraries (Java, Perl, Python, C++) for net programming.