Searching The Web

- Challenges
- Search Engine Architecture
- Crawling
- Indexing
- Ranking
- Meta Search
Challenges

- Distributed data
  more than 2.4 millions web servers
- Massive data
  over a billion pages and 10 terabytes (for texture data)
- Dynamic data
  23% pages changes daily and life of 50% pages is about 10 days
- Unstructured data
  texture and multimedia data
- Low data quality
  open environment and no editorial process
- Heterogeneous data
  different format and different languages
Other Challenges

- Short queries
  the average number of query terms is 1.5
- Popular queries
  many similar queries
- User communities
  users with different backgrounds
- Links
  28% of pages form a strongly connected core
  22% of pages point to the core
  22% of pages are pointed by the core
Functional Modules of Search Engine

- **Crawler (Spider)**
  
browse the web to gather web documents

- **Indexer**
  
index web documents

- **Search**
  
receive and fill search requests from users

- **Ranking**
  
sort the results according to their importance
Search Engine Architecture
Web Crawling

• Gather web documents for web search engines
every search engine must have a crawler

• Get information from the other parts of an organization
it may be easier to get info yourself than to ask others to give it to you

• Get information from external sites
from competitors, media web sites ...

• Build a web site for a particular domain
e.g., K-12 education, NBA, ...
Simple Crawler Algorithm

1. initialize the URL queue with an initial set of URLs
2. pop a URL from the queue
3. retrieve the page of the URL
4. parse the page to find all URLs in the page
5. discard the URLs that do not satisfy the requirements
6. add the rest of URLs to the queue
7. if not time to stop, go back to 2.
A Simple Crawler Architecture
Main Process

- coordinate the behavior of the multiple, multi-threaded crawlers
- maintain a DB of unexamined URLs
- maintain a DB of examined URLs
- distribute unexamined URLs to crawler processes
- receive UTLs from crawlers
Crawler Processes

• coordinate the behavior of the multiple downloading threads
  – network is slow and unreliable

• receive URLs from downloading threads

• return URLs to the main process
Downloading Threads

• download a page
• extract URLs from retrieved pages extract URLs from the retrieved page
• discard URLs that do not satisfy the requirements
• store the downloaded document to the document DB
• return URLs to the crawler process
Characteristics

• Robust
  – if a thread crashes or hangs, its UTL is redistributed

• Easy to restart
  – doesn’t revisit the pages downloaded

• Search strategies are easy to change
  – main process controls the crawling behavior

• Relative modular
Google Crawler (early version)

- a single URL server
- a set of crawlers
  - example: 3
- each crawler manages a set of open connections in parallel
  - example: 300
- peak speeds
  - more than 100 pages per second (4 crawlers)
  - 600k data per second
Search Strategies

- **Depth First**
  - produces a narrow crawl
  - visits pages deeply buried within each site.

- **Breath First**
  - produces a broad crawl
  - visits as many web sites as possible
  - common choice, because of its simplicity

- **Site-based Breath First**
  - search a set of given (user provided) web sites first
Other Search Strategies

• Topic Directed
  – find pages related to a topic
  – define a topic by a set of example pages or queries
  – downloaded pages similar to the examples pages

• Link Directed
  – keeps track of the links in and out of each page
  – pages with many incoming links visited first
    favors highly referenced pages
  – pages with many outgoing links visited first
    favors pages that reference many other pages
Search Strategy Restrictions

- Site-based Restrictions
  - the total number of pages downloaded in one site
  - depth from the root page

- File Type Restrictions
  - .exe, .gif, .jpg, .ps, .pdf, .doc, .ppt, .xls, .gz, .zip
Issues of Web Crawlers

- What pages should the crawler download?
  - the crawler cannot download all pages on the web
- How should the crawler refresh pages?
  - web pages change at very different rates
- How should the load on the visited web sites be minimized
  - crawler consumes web server time
- How should the load on the network be minimized
  - crawler increases network traffic
- How should the crawling process be paralleled
  - crawlers often run on multiple machines and work in parallel
Page Selection

- Importance Metrics
  - interest driven
    users provide queries and pages similar to queries are interesting
  - popularity driven
    pages with many incoming links are popular
  - location driven
    .com may be more useful than .edu
    URLs contain “home” may of more interest than other URLs
Crawl and Stop

• important or interesting pages are visited first

• stop conditions
  – number of pages to visit
    crawling stops after a given number of pages have been visited
  – threshold
    pages with importance larger than the threshold are downloaded
Page Refresh

- revisit all pages at the same frequency, regardless of how often they change
- periodically download the page header to see if the page has changed
  - download the pages that have changed
- revisit a page proportionally more often, as it changes more often.
- revisit often pages that are accessed frequently
Good Behavior

• A file, Robots.txt, stored in the root of a web site indicates whether crawlers are allowed on the site, or where

• Example
  – http://www.nvc.cs.vt.edu/Robots.txt

• Robots.txt is voluntary and crawlers may ignore it, but the site administrator may block it for the entire site.

• Always download this file and check it and visit the directories that are allowed
Examples of Robots.txt

- Exclude all crawlers from the entire site.
  User-agent: *
  Disallow: /

- Allow all crawlers to access the entire site
  User-agent: *
  Disallow:

- An empty Robots.txt allows the complete access for the entire site.
Examples of Robots.txt

• Exclude all crawlers from some directories
  User-agent: *
  Disallow: /cgi-bin/
  Disallow: /tmp/

• exclude a single crawler
  User-agent: BadCrawler
  Disallow: /

• allow a single crawler.
  User-agent: WebCrawler
  Disallow: