Nutch as a Web mining platform
the present and the future

Andrzej Białecki
ab@sigram.com

Intro

- Started using Lucene in 2003 (1.2-dev?)
- Created Luke – the Lucene Index Toolbox
- Nutch, Lucene committer, Lucene PMC member
- Nutch project lead

Agenda

- Nutch architecture overview
- Crawling in general – strategies and challenges
- Nutch workflow
- Web data mining with Nutch

  with examples 
- Nutch present and future
- Questions and answers

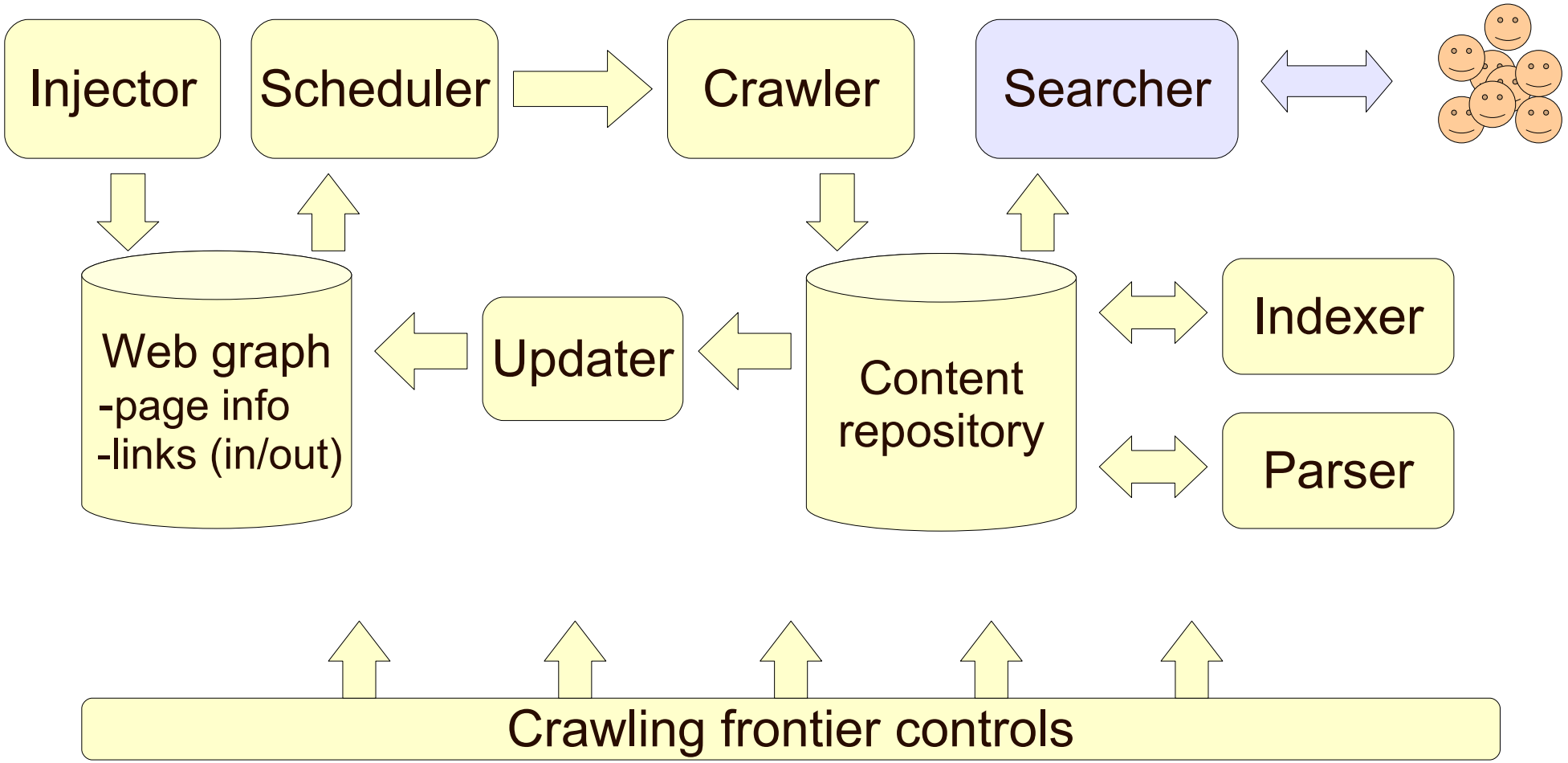
Apache Nutch project

- Founded in 2003 by Doug Cutting, the Lucene creator, and Mike Cafarella
- Apache project since 2004 (sub-project of Lucene)
- Spin-offs:
  - Map-Reduce and distributed FS → Hadoop
  - Content type detection and parsing → Tika
- Many installations in operation, mostly vertical search
- Collections typically 1 mln - 200 mln documents
- Apache Top-Level Project since May
- Current release 1.1

What's in a search engine?

… a few things that may surprise you! ☺

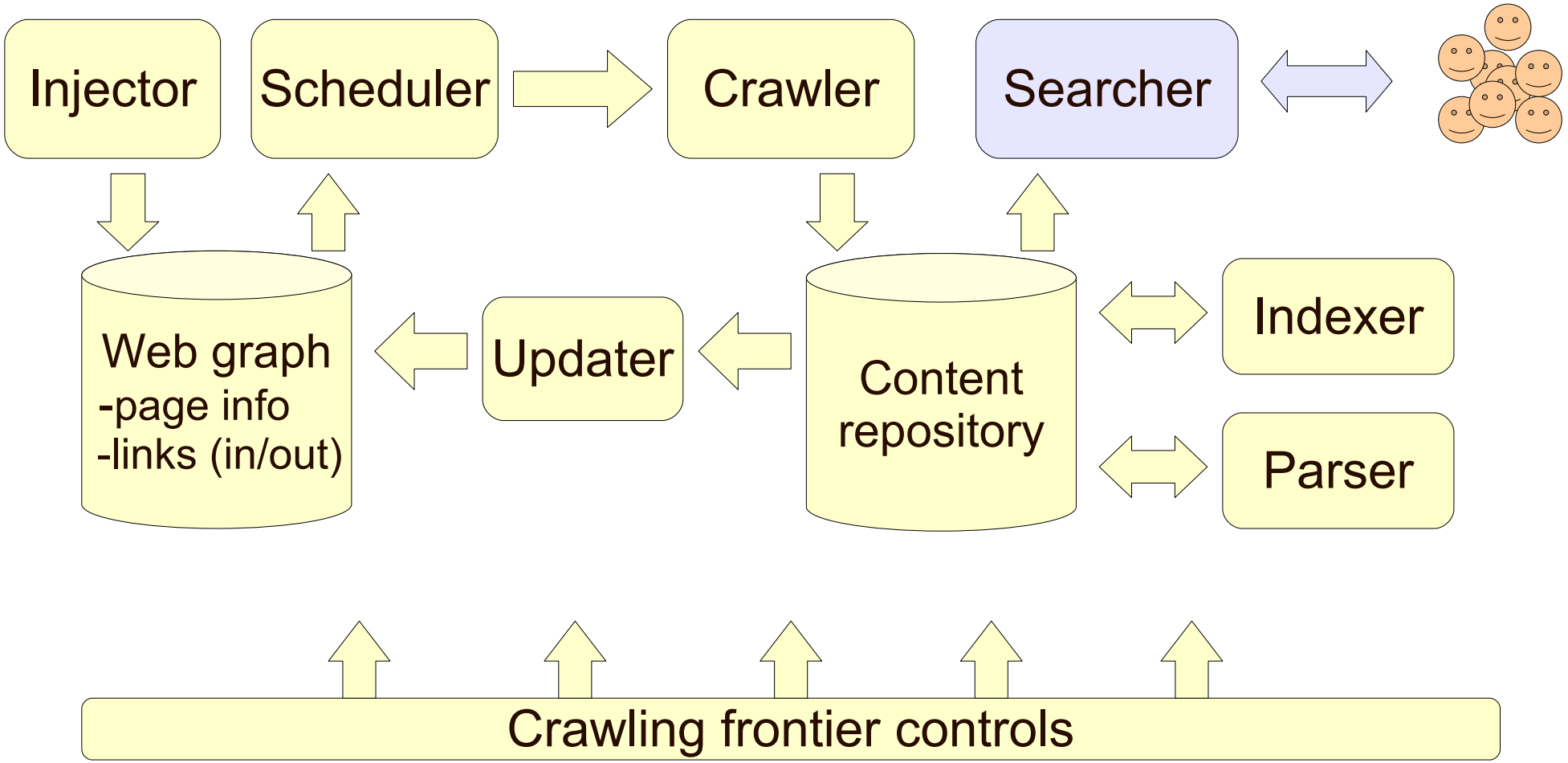
Search engine building blocks



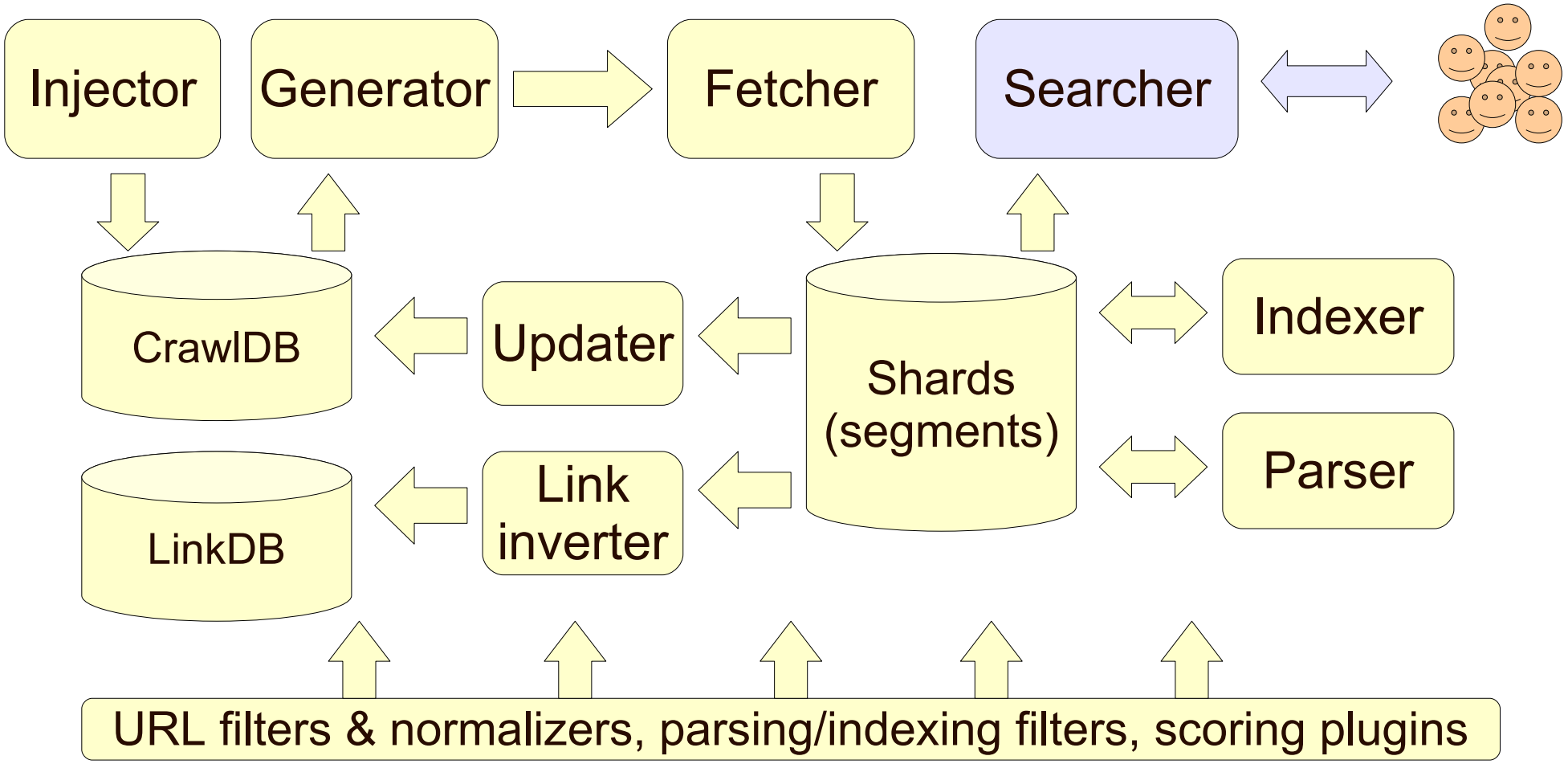
Nutch features at a glance

- Plugin-based, highly modular:
  - Most behaviors can be changed via plugins
- Data repository:
  - Page status database and link database (web graph)
  - Content and parsed data database (shards)
- Multi-protocol, multi-threaded, distributed crawler
- Robust crawling frontier controls
- Scalable data processing framework
  - Hadoop MapReduce processing
- Full-text indexer & search front-end
  - Using Solr (or Lucene)
  - Support for distributed search
- Flexible integration options

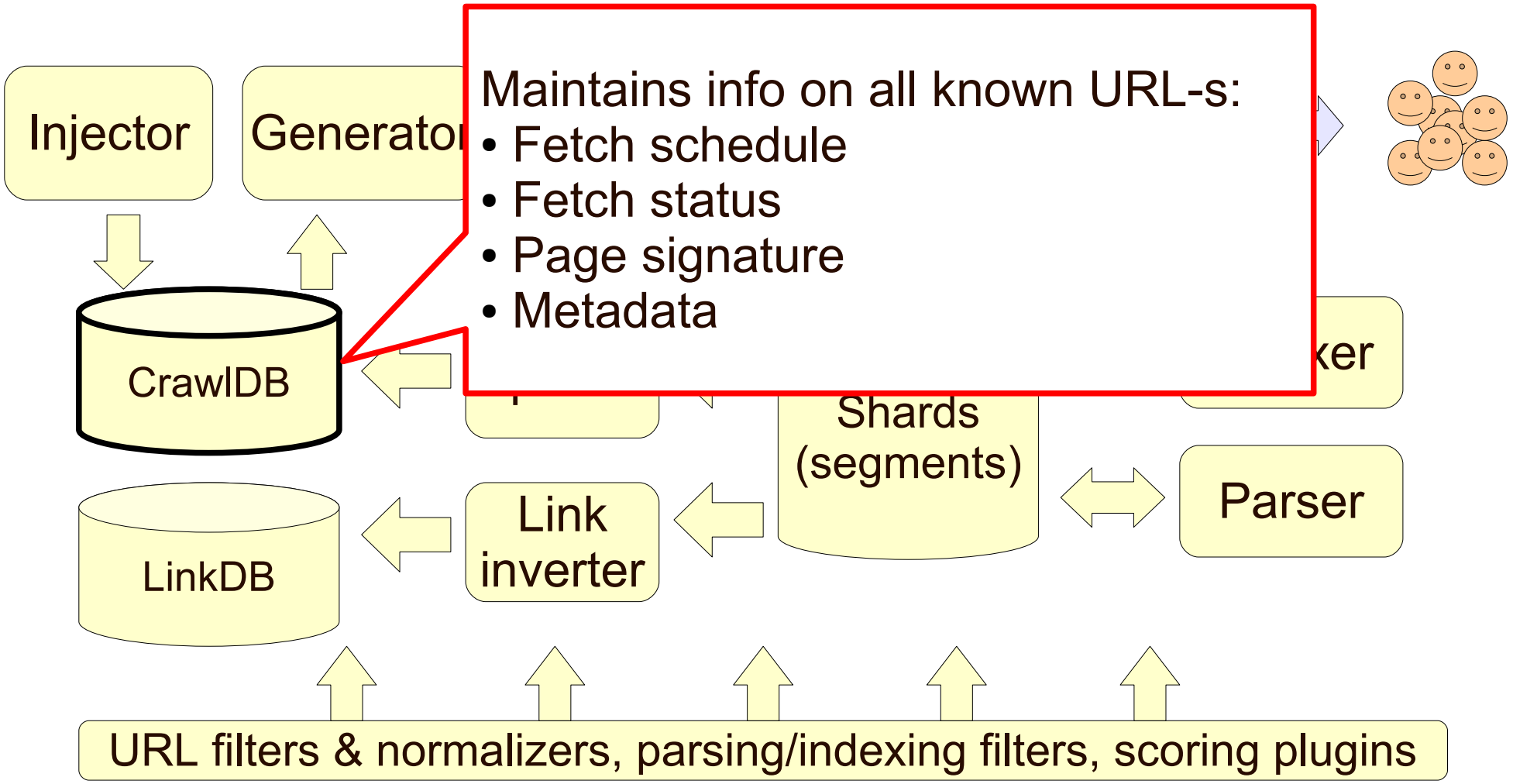
Search engine building blocks



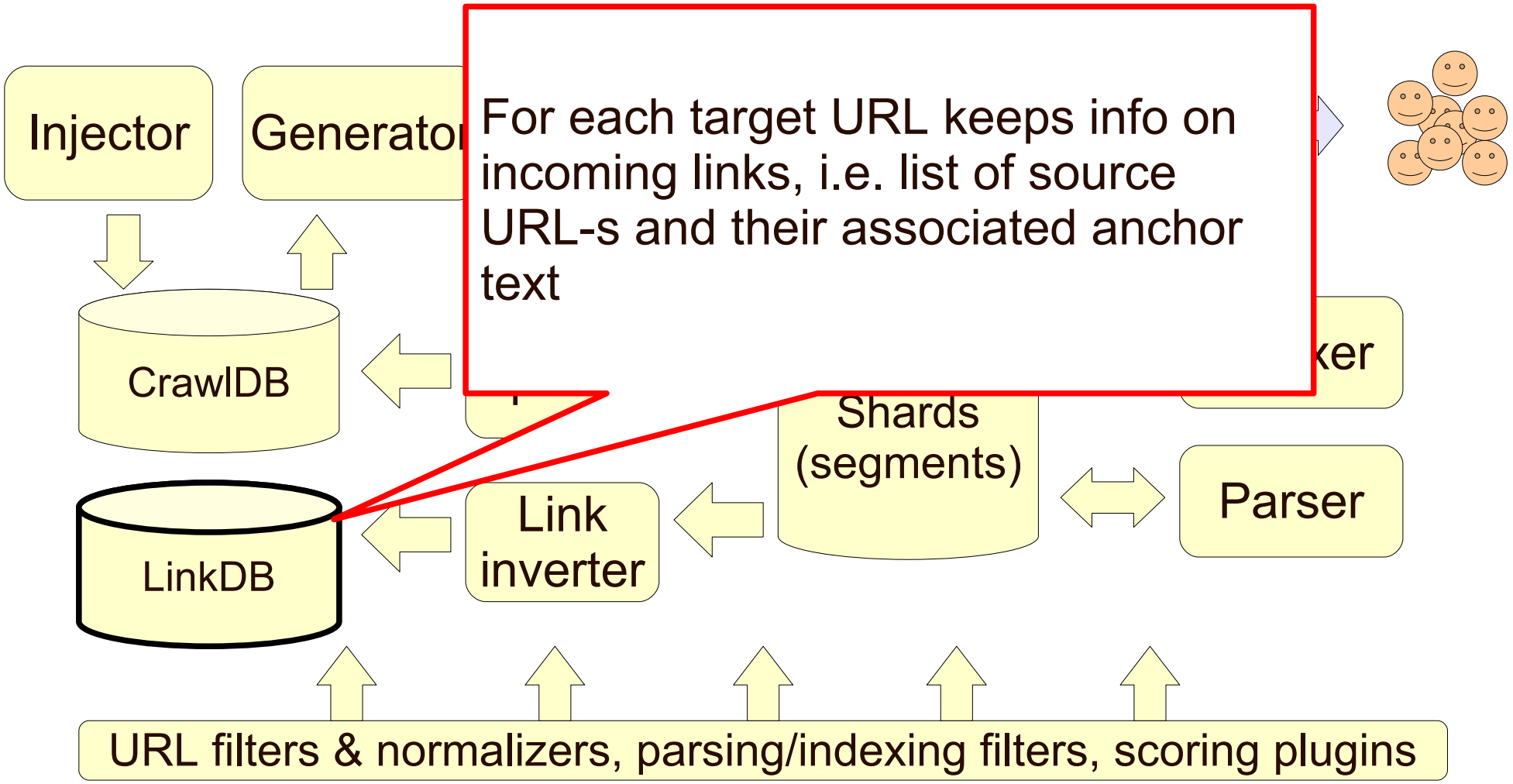
Nutch building blocks



Nutch data

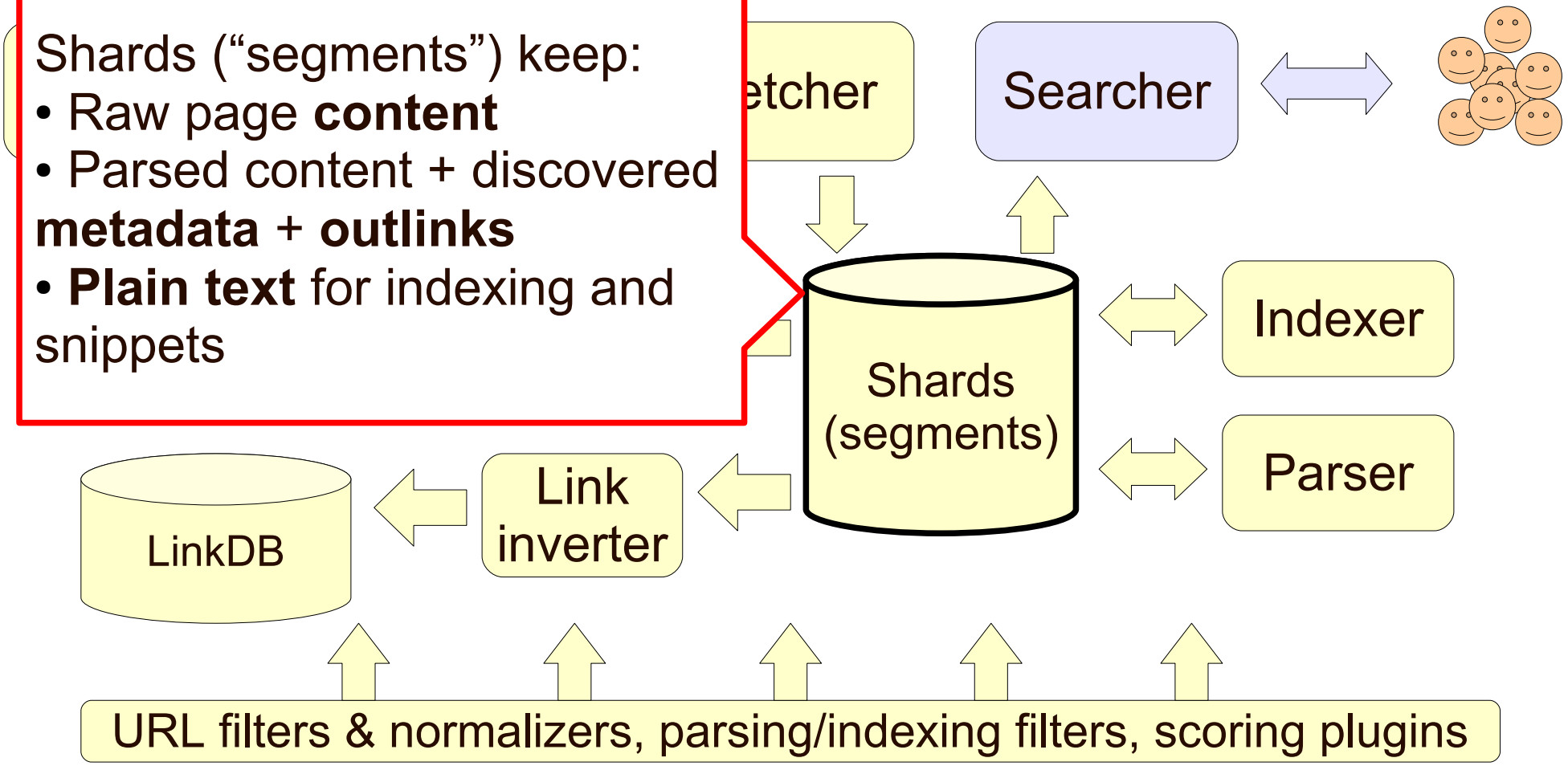

Maintains info on all known URL-s:
• Fetch schedule
• Fetch status
• Page signature
• Metadata
Injector
Generator
CrawlDB
LinkDB
Link inverter
Shards (segments)
Parser
URL filters & normalizers, parsing/indexing filters, scoring plugins

Nutch data


Injector
Generator
For each target URL keeps info on incoming links, i.e. list of source URL-s and their associated anchor text
CrawlDB
LinkDB
Link inverter
Shards (segments)
Parser
URL filters & normalizers, parsing/indexing filters, scoring plugins

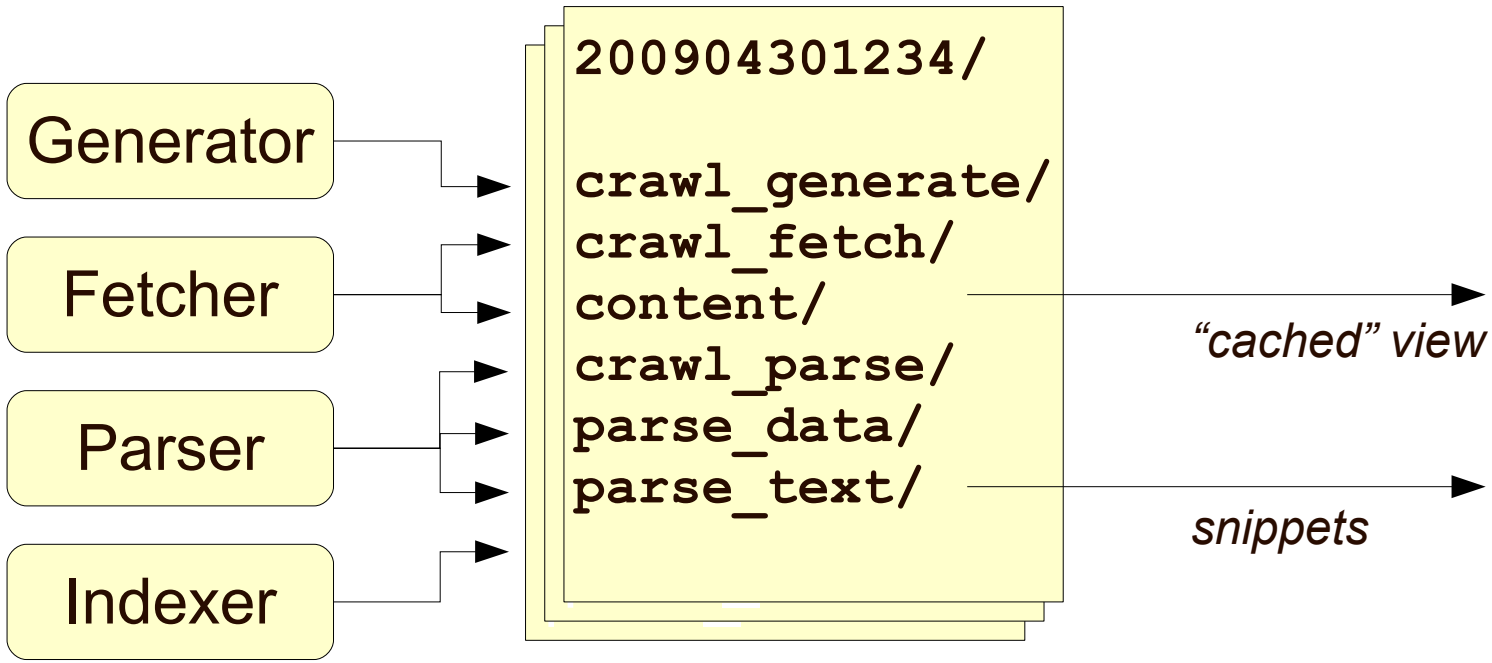
Nutch data

Shards ("segments") keep:
- Raw page **content**
- Parsed content + discovered **metadata** + **outlinks**
- **Plain text** for indexing and snippets



Shard-based workflow

- Unit of work (batch) – easier to process massive datasets
- Convenience placeholder, using predefined directory names
- Unit of deployment to the search infrastructure
  - Solr-based search may discard shards once indexed
- Once completed they are basically unmodifiable
  - No in-place updates of content, or replacing of obsolete content
- Periodically phased-out by new, re-crawled shards
  - Solr-based search can update Solr index in-place



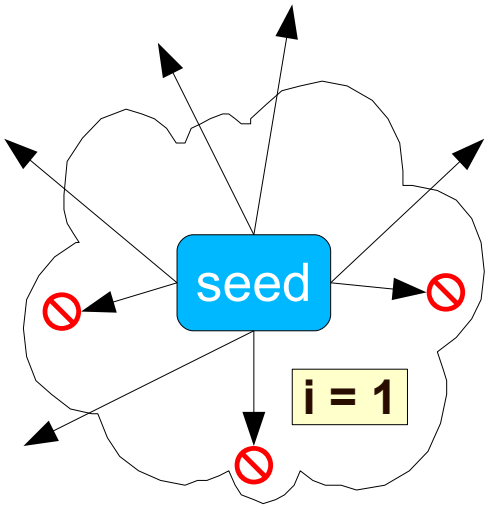
Crawling frontier challenge

- No authoritative catalog of web pages
- Crawlers need to discover their view of web universe
  - Start from “seed list” & follow (walk) some (*useful? interesting?*) outlinks
- Many dangers of simply wandering around
  - explosion or collapse of the frontier; collecting unwanted content (spam, junk, offensive)

High-quality seed list

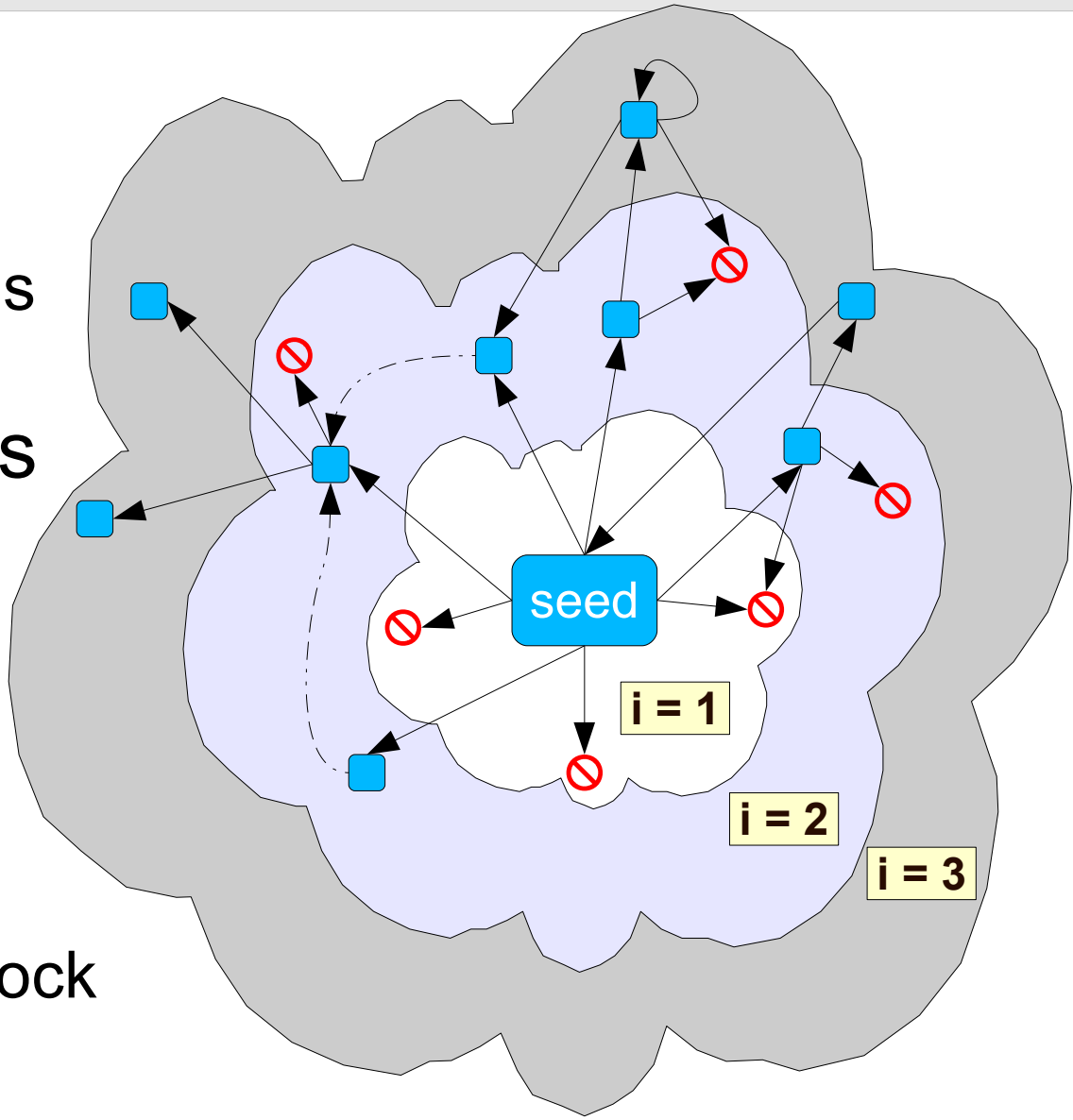
- Reference sites:
  - Wikipedia, FreeBase, DMOZ
  - Existing verticals
- Seeding from existing search engines
  - Collect top-N URL-s for characteristic keywords
- Seed URL-s plus 1:
  - First hop usually retains high-quality and focus
  - Remove blatantly obvious junk

seed + 1 hop



Controlling the crawling frontier

- URL filter plugins
  - White-list, black-list, regex
  - May use external resources (DB-s, services ...)
- URL normalizer plugins
  - Resolving relative path elements
  - “Equivalent” URLs
- Additional controls
  - priority, metadata select/block
  - Breadth first, depth first, per-site mixed ...



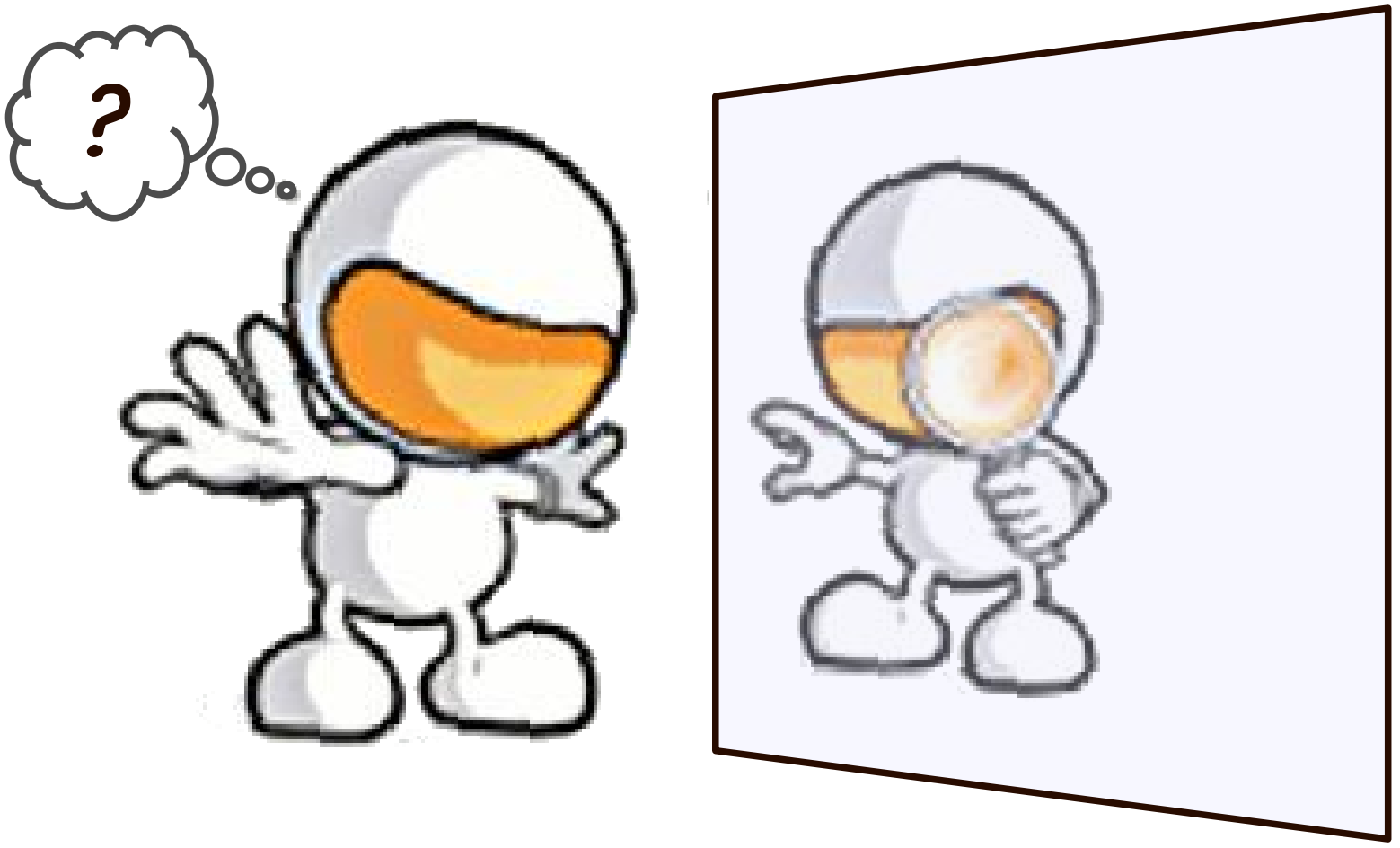
Wide vs. focused crawling

- Differences:
  - Little technical difference in configuration
  - Big difference in operations, maintenance and quality
- Wide crawling:
  - (Almost) Unlimited crawling frontier
  - High risk of spamming and junk content
  - **“Politeness” a very important limiting factor**
  - Bandwidth & DNS considerations
- Focused (vertical or enterprise) crawling:
  - Limited crawling frontier
  - Bandwidth or politeness is often not an issue
  - Low risk of spamming and junk content

Vertical & enterprise search

- Vertical search
  - Range of selected “reference” sites
  - Robust control of the crawling frontier
  - Extensive content post-processing
  - Business-driven decisions about ranking
- Enterprise search
  - Variety of data sources and data formats
  - Well-defined and limited crawling frontier
  - Integration with in-house data sources
  - Little danger of spam
  - PageRank-like scoring usually works poorly

Face to face with Nutch



Installation & basic config

- http://nutch.apache.org
- Java 1.5+
- Single-node out of the box
  - Comes also as a “job” jar to run on existing Hadoop cluster
- File-based configuration: conf/
  - Plugin list
  - Per-plugin configuration
- … much, much more on this on the Wiki

Main Nutch workflow

Command-line:
`bin/nutch`

- **Inject**: initial creation of CrawlDB
  - Insert seed URLs
  - Initial LinkDB is empty

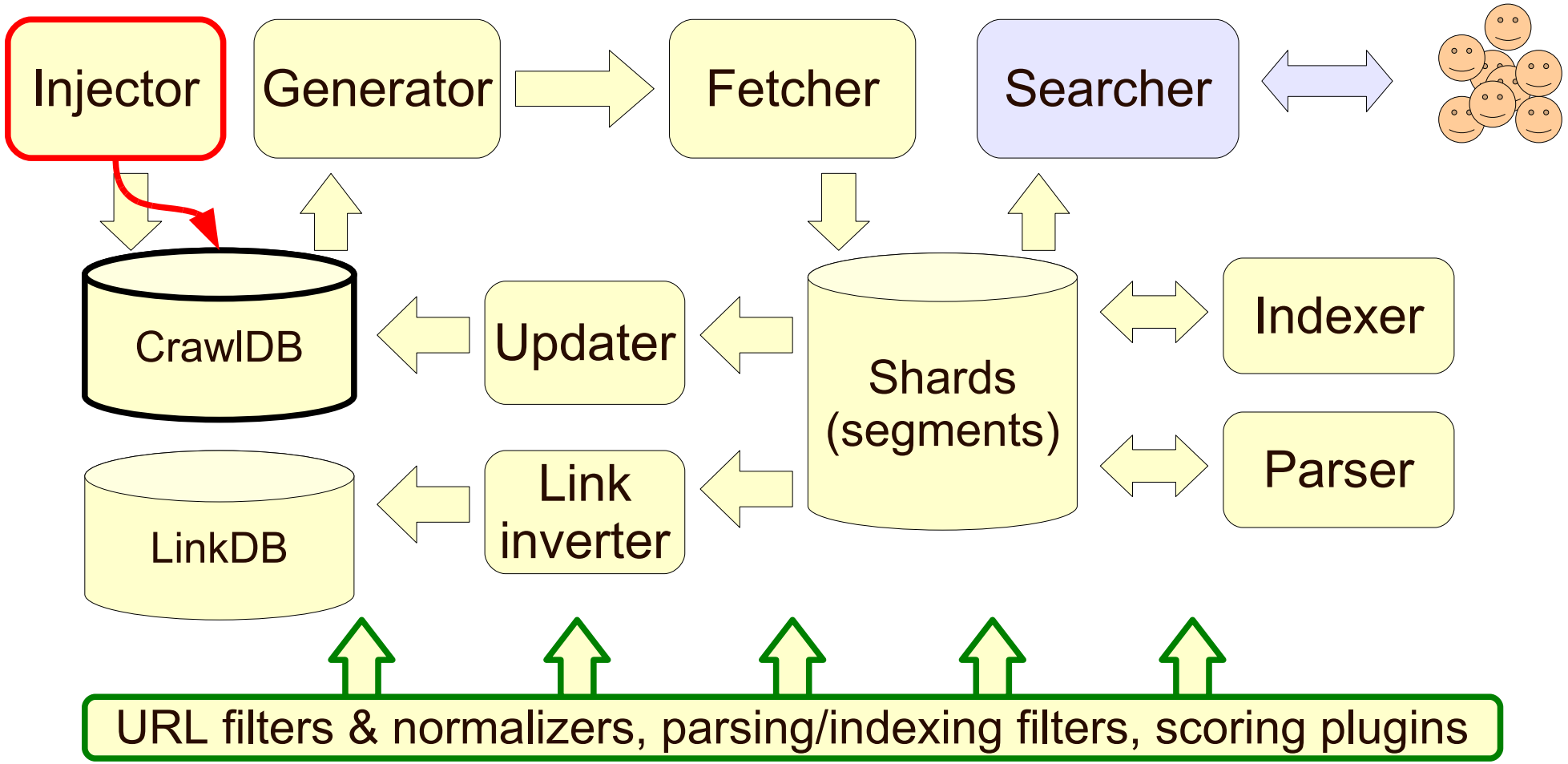
`inject`

(repeat)

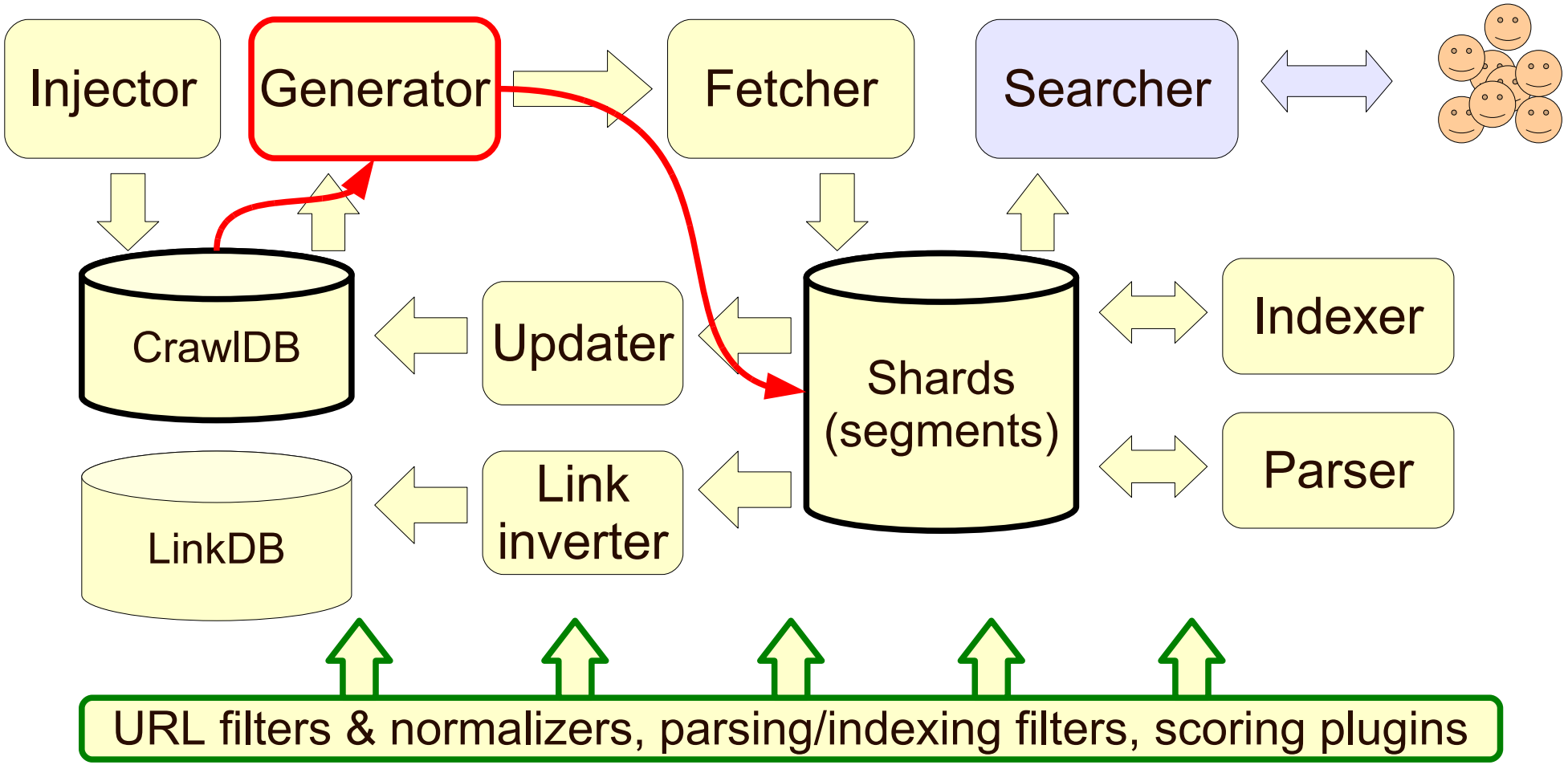
- **Generate** new shard's fetchlist
- **Fetch** raw content
- **Parse** content (discovers outlinks)
- **Update CrawlDB** from shards
- **Update LinkDB** from shards
- **Index** shards

`generate`
`fetch`
`parse`
`updatedb`
`invertlinks`
`index / solrindex`

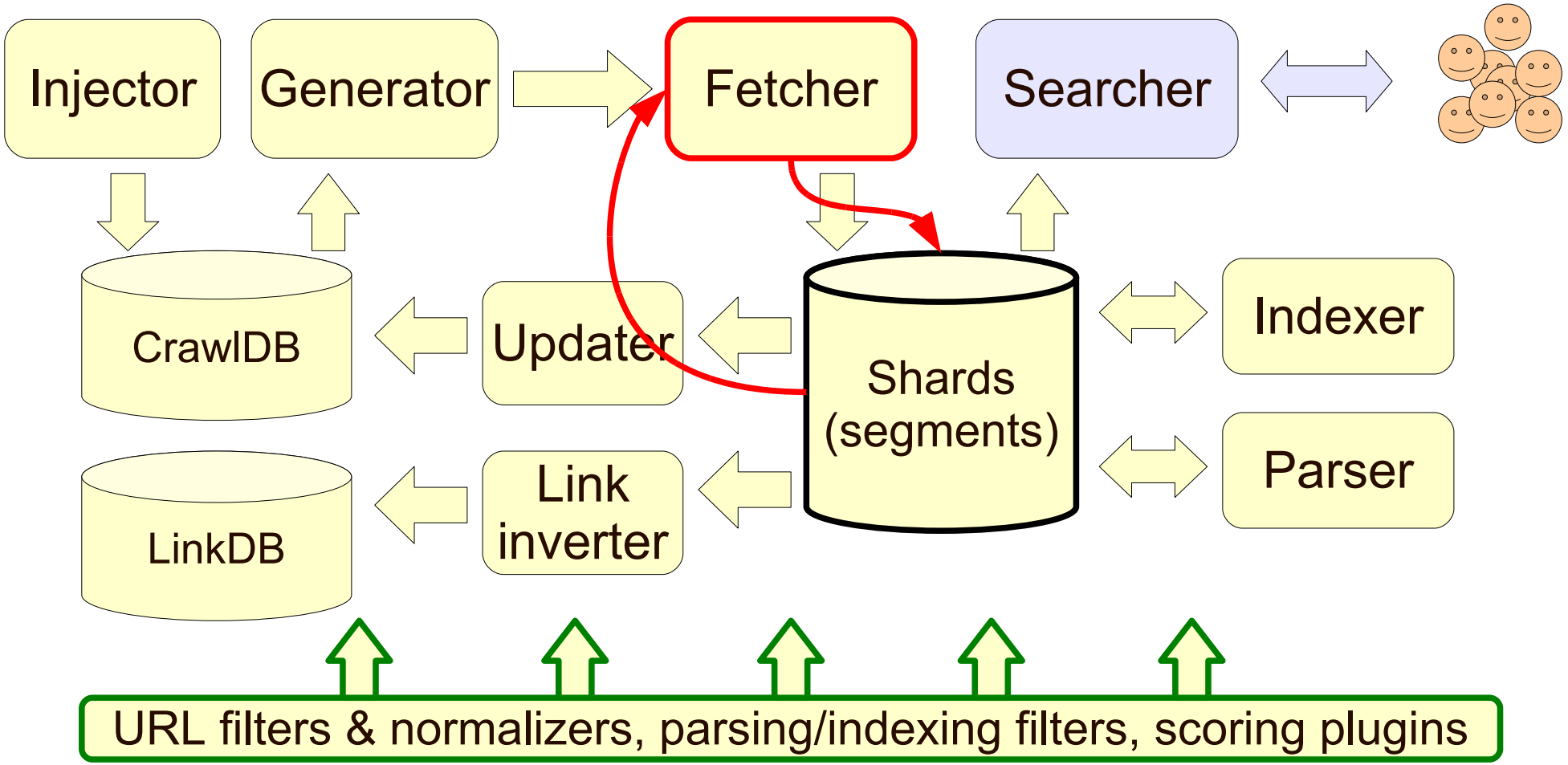
Injecting new URL-s



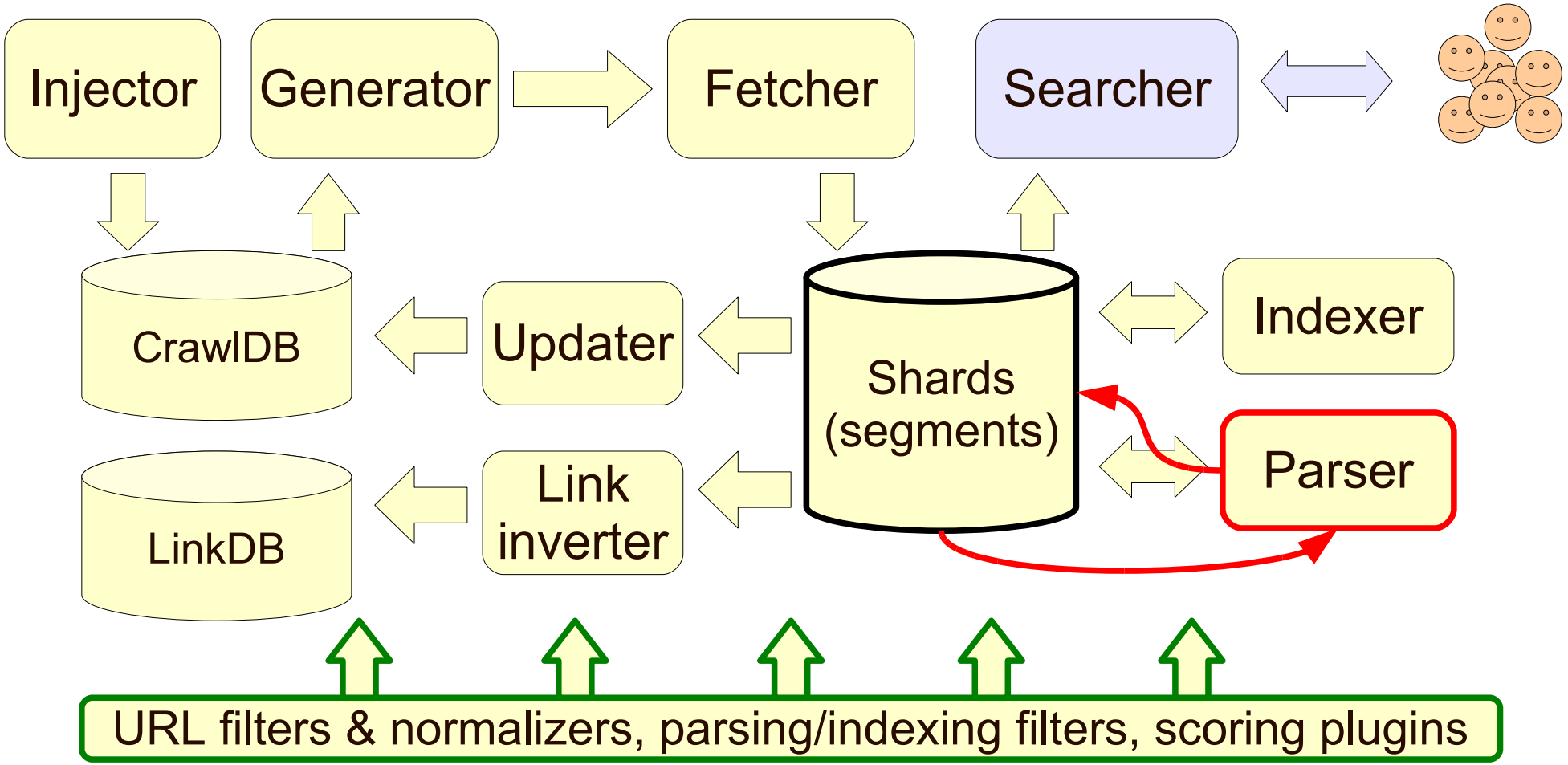
Generating fetchlists



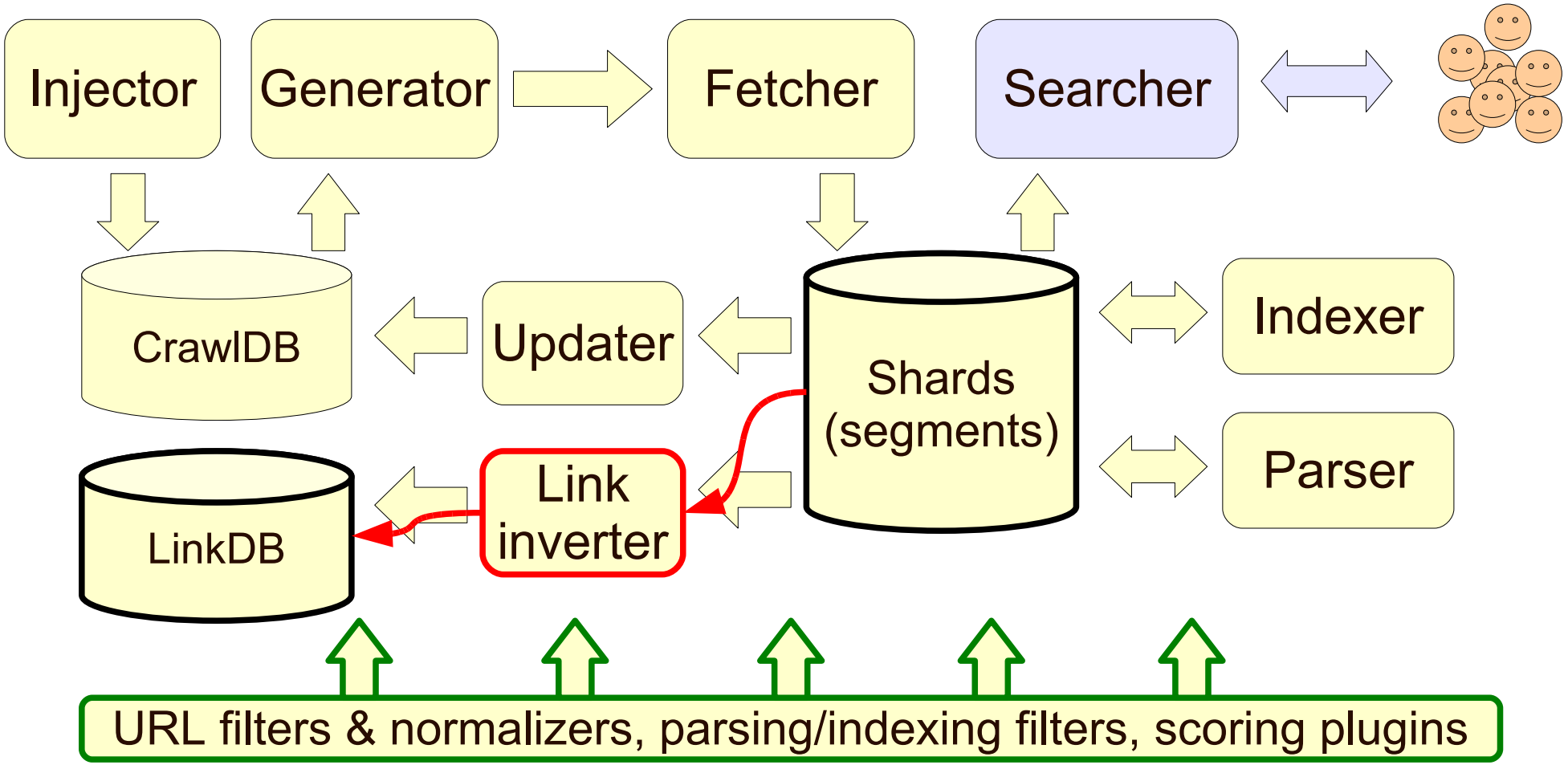
Fetching content



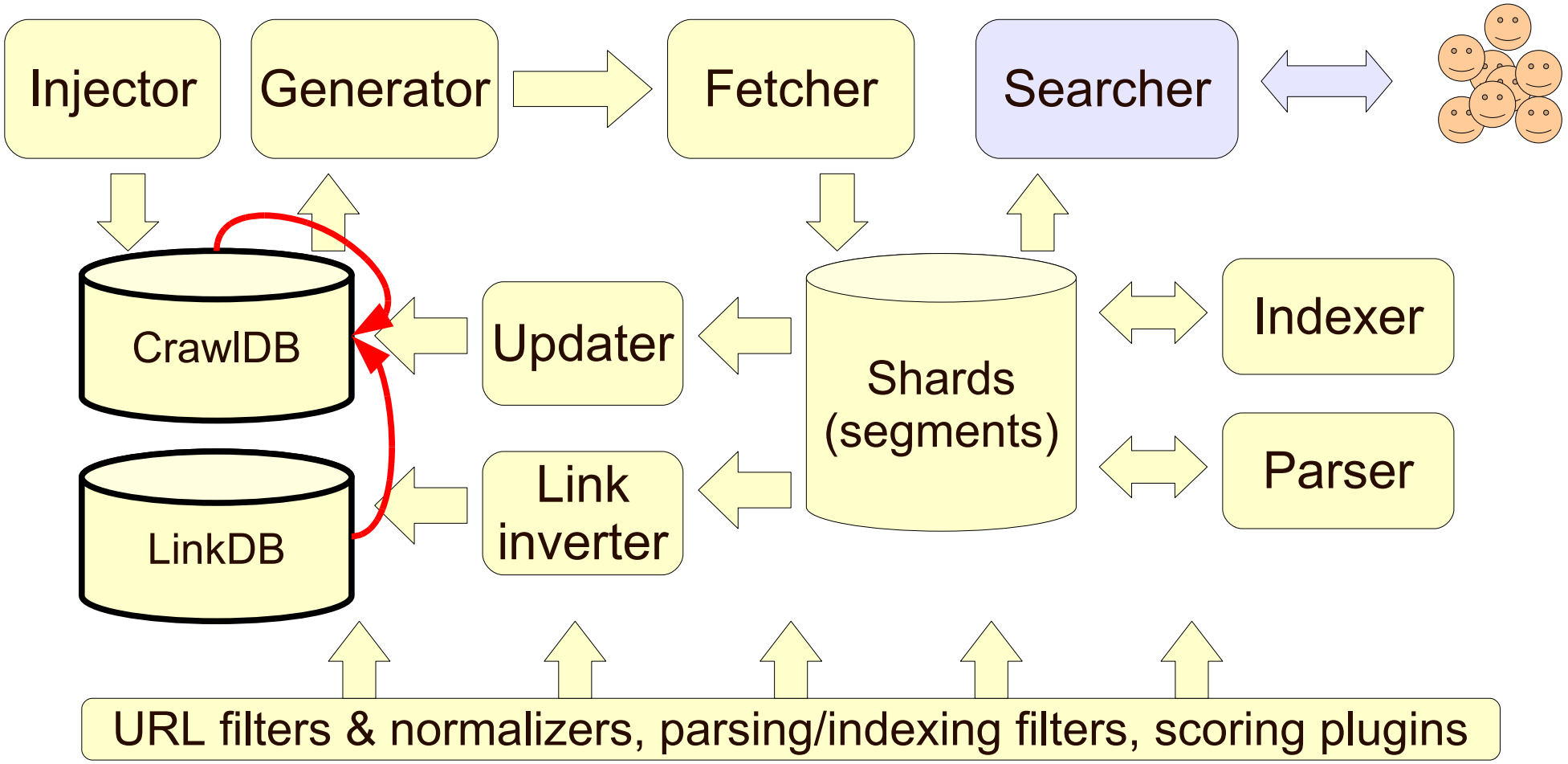
Content processing



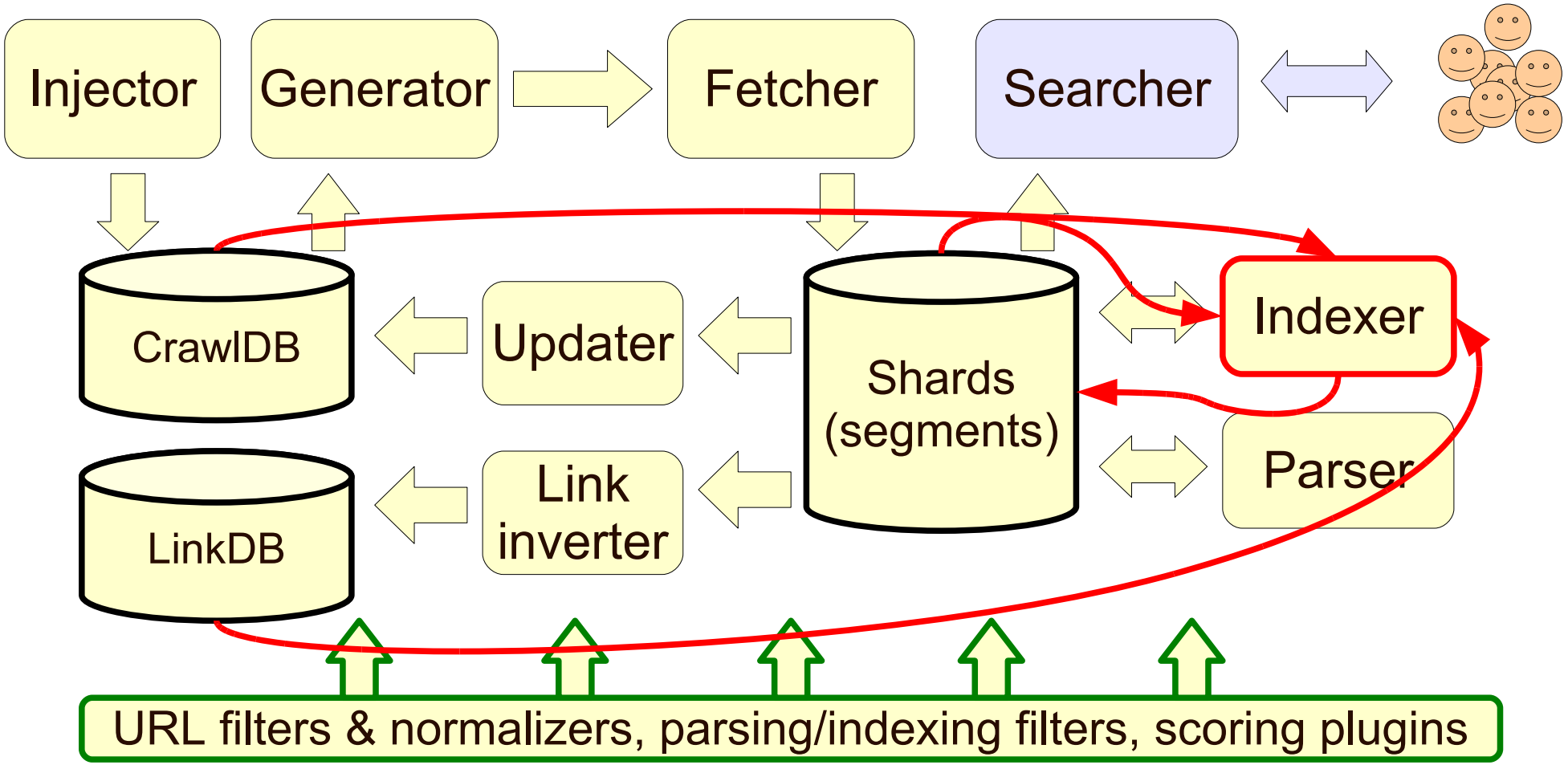
Link inversion



Page importance - scoring



Indexing


Injector
Generator
Fetcher
Searcher
CrawlDB
LinkDB
Updater
Link inverter
Shards (segments)
Indexer
Parser
URL filters & normalizers, parsing/indexing filters, scoring plugins

Map-reduce indexing

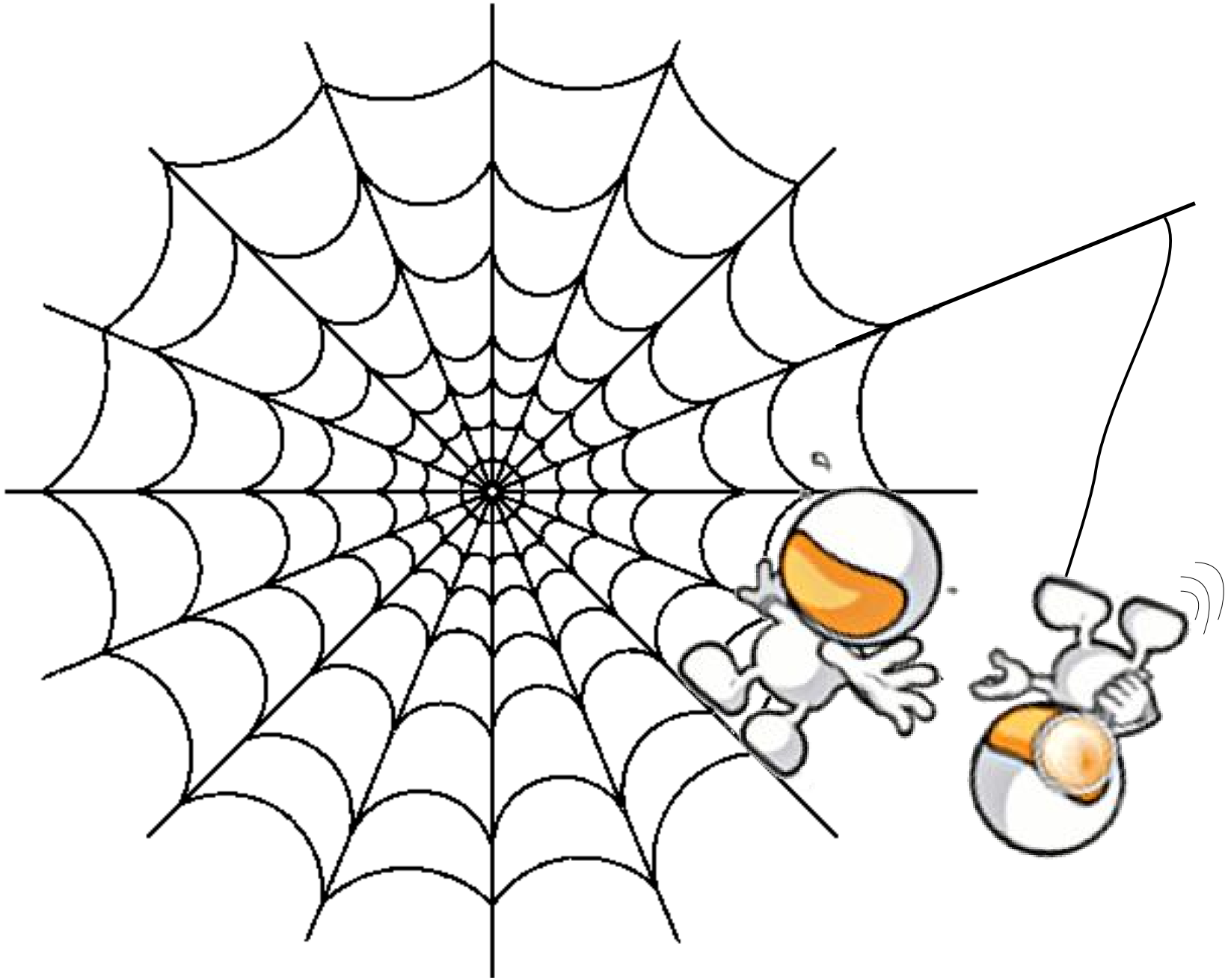
- Map() just assembles all parts of documents
- Reduce() performs text analysis + indexing:
  - Sends assembled documents to Solr
    or
  - Adds to a local Lucene index

- *Other possible MR indexing models:*
  - *Hadoop contrib/indexing model:*
    - *analysis and indexing on map() side*
    - *Index merging on reduce() side*
  - *Modified Nutch model:*
    - *Analysis on map() side*
    - *Indexing on reduce() side*

Nutch integration

- Nutch search & tools API
  - Search via REST-style interaction, XML / JSON response
  - Tools CLI and API to access bulk & single Nutch items
  - Single-node, embedded, distributed (Hadoop cluster)
- Data-level integration: direct MapFile / SequenceFile reading
  - More complicated (and still requires using Nutch classes)
  - May be more efficient
  - Future: native tools related to data stores (HBase, SQL, ...)
- Exporting Nutch data
  - All data can be exported to plain text formats
  - `bin/nutch read*`
    - `...db` – read CrawlDB and dump some/all records
    - `...linkdb` – read LinkDb and dump some/all records
    - `...seg` – read segments (shards) and dump some/all records

Web data mining with Nutch



Nutch search

- Solr indexing and searching (preferred)
  - Simple Lucene indexing / search available too
- Using Solr search:
  - DisMax search over several fields (url, title, body, anchors)
  - Faceted search
  - Search results clustering
  - SolrCloud:
    - Automatic shard replication and load-balancing
    - Hashing update handler to distribute docs to Solr shards

Search-based analytics

- Keyword search → crude topic mining
- Phrase search → crude collocation mining
- Anchor search → crude semantic enrichment
- Feedback loop from search results:
  - Faceting and on-line clustering may discover latent topics
  - Top-N results for reference queries may prioritize further crawling
- Example: question answering system
  - Source documents from reference sites
  - NLP document analysis: key-phrase detection, POS-tagging, noun-verb / subject-predicate detection, enrichment from DBs and semantic nets
  - NLP query analysis: expected answer type (e.g. person, place, date, activity, method, ...), key-phrases, synonyms
  - Regular search
  - Evaluation of raw results (further NLP analysis of each document)

Web as a corpus

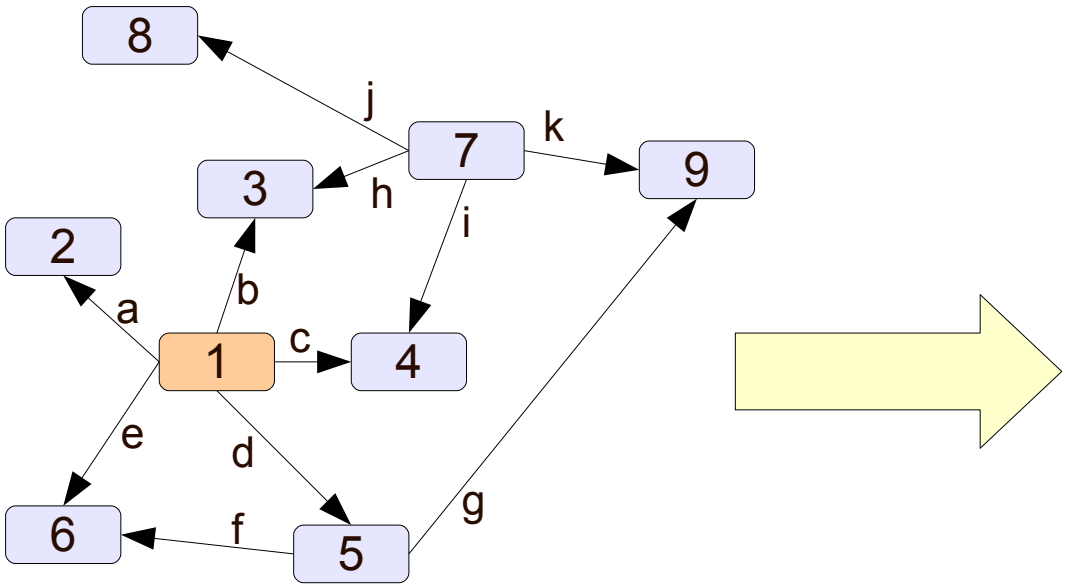
- Examples:
  - Source of raw text in a specific language
  - Source of text on a given subject
    - Selection by e.g. a presence of keywords, or full-blown NLP
    - Add data from known reference sites (Wikipedia, Freebase) or databases (Medline) or semantic nets (WordNet, OpenCyc)
  - Source of documents in a specific format (e.g. PDF)
- Nutch setup:
  - URLFilters define the crawling frontier and content types
  - Parse plugins determine the content extraction / processing
    - e.g. language detection
- Nutch shards:
  - Extracted text, metadata, outlinks / anchors

Web as a corpus (2)

- Concept mining
  - Harvesting human-created concept descriptions and associations
  - “kind of”, “contains”, “includes”, “application of”
  - Co-occurrence of concepts has some meaning too!
- Example: medical search engine
  - Controlled vocabulary of diseases, symptoms, procedures
  - Identifiable metadata: author, journal, publication date, etc.
  - Nutch crawl of reference sites and DBs
    - Co-occurrence of controlled vocabulary
      - BloomFilter-s for quick trimming of map-side data
      - Or Mahout collocation mining for uncontrolled concepts
    - Cube of co-occurring (related) concepts
    - Several dimensions to traverse
      - “authors who publish most often together on treatment of myocardial infarction”
    - 10 nodes, 100k phrases in vocabulary, 20 mln pages, ~300bln phrases on map side → ~5GB data cube

Web as a directed graph

- Nodes (vertices): URL-s as unique identifiers
- Edges (links): hyperlinks like <a href="targetUrl"/>
- Edge labels: anchor text
- Often represented as adjacency (neighbor) lists
- Inverted graph: LinkDB in Nutch

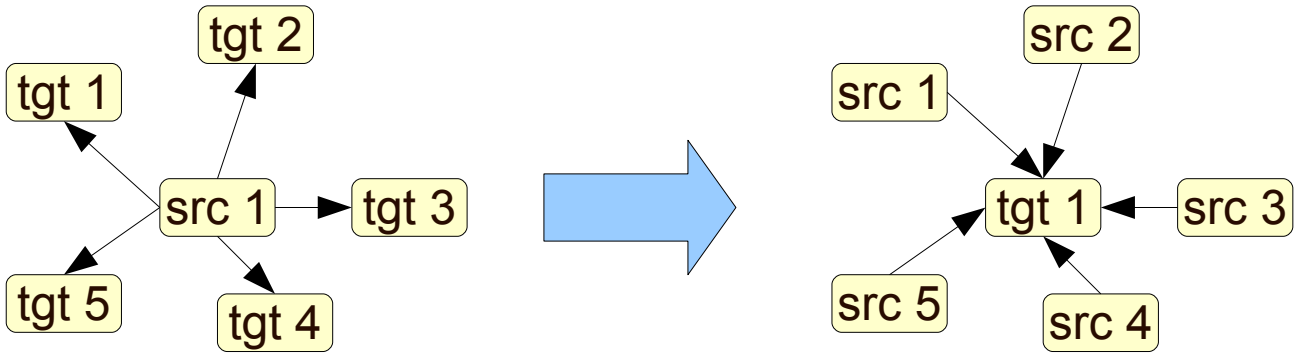


Straight (outlink) graph:
1 → 2a, 3b, 4c, 5d, 6e
5 → 6f, 9g
7 → 3h, 4i, 8j, 9k

Inverted (inlink) graph:
2 ← 1a
3 ← 1b, 7h
4 ← 1c, 7i
5 ← 1d
6 ← 1e, 5f
8 ← 7j
9 ← 5g, 7k

Link inversion

- Pages have outgoing links (outlinks)
  - … I know where I'm pointing to
- **Question**: who points to me?
  - … I don't know, there is no catalog of pages
  - … NOBODY knows for sure either!
- In-degree may indicate importance of the page
- Anchor text provides important semantic info
- **Answer**: invert the outlinks that I know about, and group by target (Nutch 'invertlinks')

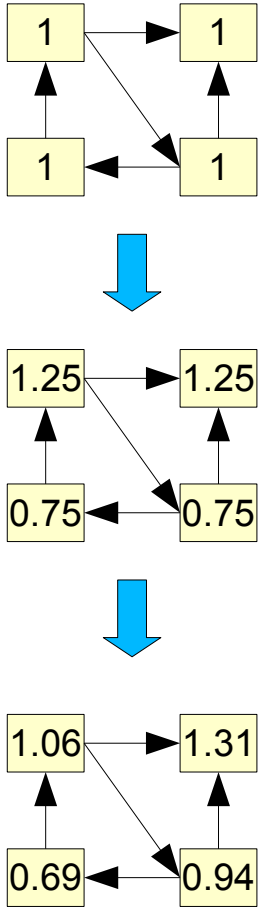


Web as a recommender

- Links as recommendations:
  - Link represents an association
  - Anchor text represents a recommended topic
    - … with some surrounding text of a hyperlink?
- Not all pages are created equal
  - Recommendations from good pages are useful
  - Recommendations from bad pages may be useless
  - Merit / guilt by association:
    - Links from good pages should improve the target's reputation
    - Links from bad pages may compromise good pages' reputation
- Not all recommendations are trustworthy
  - What links to trust, and to what degree?
  - Social aspects: popularity, fashion, mobbing, fallacy of "common belief"

Link analysis and scoring

- PageRank
  - Query-independent page weight
  - Based on the flow of weight along link paths
    - Dampening factor α to stabilize the flow
    - Weight from “dangling nodes” redistributed
- Other models
  - Hyperlink-Induced Topic Search (HITS)
    - Query-dependent, local iterations, hub/authority
  - TrustRank
    - Propagation of “trust” based on human expert evaluation of seed sites
- Challenges
  - Loops, link spam, cliques, loosely connected subgraphs, mobbing, etc



Nutch link analysis tools

- Tools for PageRank calculation with loop detection
  - LinkDb: source of anchor text (think “recommended topics”)
  - Page in-degree ≈ popularity / importance / quality
  - Scoring API (and plugins) to control the flow of page importance along link paths
- Nutch shards:
  - Source of outlinks → expanding the crawling frontier
  - Page linked-ness vs. its content: hub or authority
- Example: porn / junk detection
  - Links to “porn” pages poisonous to importance / quality
  - Links from “porn” pages decrease the confidence in quality of the target page
- Example: vertical crawl
  - Expanding to pages “on topic” == with sufficient in-link support from known on topic pages

Web of gossip and opinions

- General Web – not considering special-purpose networks here...
- Example:
  - Who / what is in the news?
  - How often a name is mentioned?
    - today Google yields 44,500 hits for ab@getopt.org ☺
  - What facts about me are publicly available?
  - What is the sentiment associated with a name (person, organization, trademark)?
- Nutch setup:
  - Seed from a few reference news sites, blogs, Twitter, etc
  - Use Nutch plugin for RSS/Atom crawling
  - NLP parsing plugins (NER, classification, sentiment analysis)
- Nutch shards:
  - Capture temporal aspect

Web as a source of … anything

- The data is there, just lost among irrelevant stuff
  - Difficult to find → good seed list + crawling frontier controls
  - Mixed with junk & irrelevant data → URL & content filtering

- Be creative – combine multiple strategies:
  - Crawl for raw data, stay on topic – filter out junk early
  - Use plain indexing & search as a crude analytic tool
  - Use creative post-processing to filter and enhance the data
  - Export data from Nutch and pipe it to other tools (Pig, HBase, Mahout, ...)

Future of Nutch

- Nutch 2.0 re-design
  - Refactoring, cleanup, better scale-up / scale-down
  - Avoid code duplication
  - Expected release ~Q4 2010
- Share code with other crawler projects → crawler-commons
- Indexing & Search → Solr, SolrCloud
  - Distributed and replicated search is difficult
  - Initial integration needs significant improvement
  - Shard management – SolrCloud / Zookeeper
- Web-graph & page repository → ORM layer
  - Combine CrawlDB, LinkDB and shard storage
  - Avoid tedious shard management
  - Gora ORM mapping: HBase, SQL, Cassandra? BerkeleyDB?
  - Benefit from native tools specific to storage → easier integration

Future of Nutch (2)

- What's left then?
  - Crawling frontier management, discovery
  - Re-crawl algorithms
  - Spider trap handling
  - Fetcher
  - Ranking: enterprise-specific, user-feedback
  - Duplicate detection, URL aliasing (mirror detection)
  - Template detection and cleanup, pagelet-level crawling
  - Spam & junk control
- Vision: á la carte toolkit, scalable from 1-1000s nodes
  - Easier setup for small 1 node installs
  - Focus on a reliable, easy to integrate framework

Conclusions

(This overview is a tip of the iceberg)

Nutch

- Implements all core search engine components
- Extremely configurable and modular
- Scales well
- A complete crawl & search platform – and a toolkit
- Easy to use as an input feed to data collecting and data mining tools

Q & A

- Further information:
  - http://nutch.apache.org/
  - user@nutch.apache.org
- Contact author:
  - ab@sigram.com