The Web:

- a source of distributed information
- a giant set of semi-structured data

⇒ search engines are invaluable to locate information
Motivation

- up-to-date index database
  └──
- efficient traversal
  └──
- parallelization
  └──
- distributed architecture
  └──
- increased complexity
Objectives

- scalability
- easy configuration and management
- support for extension
- robustness, resilience to failures
Architectural overview

Two separate layers:

**Component framework**
- General tasks
  - component interaction
  - lifecycle management
  - transparent interprocess communication

**Crawling application**
- Field-specific issues
  - downloading documents
  - extracting hyperlinks
  - administering page references
  - scheduling requests
Design

- the component framework exposes general component skeletons that realize common behavior
- new, field-specific components are created by means of inheritance
- the framework provides loose coupling between components

Advantages:

+ simpler and faster development
+ openness for extension
Building blocks of the architecture

- **Components**
  encapsulate field-specific functionality, produce, consume or transform data

- **Providers**
  give access to data sources

- **Connectors**
  provide asynchronous, message-based communication between components
Components

- abstract base class implements generic tasks
- differentiated subclasses based on how they interact with environment
**Components**

**Introduction**

- Component framework
- Building blocks
  - Components
  - Providers
  - Connectors
- Crawler application
- Conclusions

A component-based distributed framework for web traversal:

- Component framework
  - GenericComponent
    - Generic-Producer
    - Generic-Consumer
    - Simple-Filter
    - Complex-Filter
      - Synchronous-OutputFilter
      - Asynchronous-ComplexFilter
      - Synchronous-ComplexFilter
      - SemiSynchronous-ComplexFilter
Providers

- wrap external resources used by components
- synchronized access to data sources
- diverse functionality:
  - access databases
  - transparent cache mechanisms
  - network resources
Connectors

- abstractions of typed queues
- represent a message queue
- intra-process or inter-process
- support one-to-many, many-to-many relationships, identification by roles
Relization of connectors

Method of message transfer transparent to components:

- **Local connector**
  - typed FIFO queue
  - data is passed *by reference*

- **Remote connector**
  - corresponds to two local queues and associated network communication components in separate processes
  - data is *serialized* (and transmitted over TCP)
Client-server architecture:

- **clients** retrieve documents with respect to the appropriate traversal strategy
- the **server** partitions the web and assigns partitions to clients

Implementation using component framework classes
Marshaler component

- forwards incoming URLs to clients based on domain or host name
- caches recently forwarded URLs to decrease network load
Limited data exchange during web traversal:

- *locality principle*: approx. 10% of hyperlinks are outbound from host or domain
- *batch transmission*
- *Zipfian distribution*: discarding cached URLs leads to sharply reduced load

Load balancing between marshalers: URL distribution based on URL host name hash
Basic client components

Introduction

Component framework

Crawler application

Architecture

Server components

Client components

Traversal

Load balancing

Parsing

URL distributor component

Conclusions
**Traversing component**

**Server**
- URL belonging to Client 1
- external url
- url belonging to Client 1
- base url, links

**URL distributor**
- local url queue
- internal url
- next url
- host, #new items
- url, length, start/stop time, HTTP status code
- url, HTTP header, document content

**Parser**
- document url, referrer url
- finished

**Downloader**

---

**Components**
- Introduction
- Component framework
- Crawler application
- Architecture
  - Server components
  - Client components
- Traversal
  - Load balancing
  - Parsing
  - URL distributor component
- Conclusions
Crawler.NET

A component-based distributed framework for web traversal

Introduction

Component framework

Crawler application

Architecture

Server components

Client components

Traversal

Load balancing

Parsing

URL distributor component

Conclusions

- fetches new URLs to download from *persistent storage*

- *notification* on arrival of new URLs from server or availability of a host

- selects next URL based on traversal strategy (breadth-first, relevance-based, etc.)
Load balancing component

- Prevents overloading hosts
- Cooperates with traversal components
- Configurable delay between requests
- Dynamic adaptation based on response times
Crawler.NET
A component-based distributed framework for web traversal

Load balancing component

Introduction
Component framework
Crawler application
Architecture Server components
Client components
Traversal
Load balancing
Parsing
URL distributor component
Conclusions
Crawler.NET
A component-based distributed framework for web traversal

Parser component

Introduction
Component framework
Crawler application
Architecture
Server components
Client components
Traversal
Load balancing
Parsing
URL distributor component
Conclusions

Server
- url belonging to Client 1

URL distributor
- external url
- internal url
- local url queue
- base url, links

Traversal component
- next url
- host, #new items
- url, length, start/stop time, HTTP status code
- document url, referrer url

Downloader
- url, HTTP header, document content

Client 1
- finished

Parser
- document url, referrer url
Parser component

Extracts hyperlinks from documents:

- nonstandard HTML files
- versatile hyperlink manifestations
- performance issues (e.g. interpreting scripts)
- document types and document encoding
Crawler.NET

A component-based distributed framework for web traversal

Introduction

Component framework

Crawler application

Architecture

Server components

Client components

Traversal

Load balancing

Parsing

URL distributor component

Conclusions

Server

Traversal component

Downloader

Parser

URL distributor

Conclusions

Client 1

local url queue

base url, links

url belonging to Client 1

document url, referrer url

url, length, start/stop time, HTTP status code

host, #new items

next url

internal url

external url

url, HTTP header, document content

finished
URL distributor component

- configurable URL-patterns: what to keep and what to omit
- robot exclusion rules
- crawler traps
Future work

- configuration and monitoring from graphical user interface
- dynamic, run-time re-configuration
- dedicated data structures instead of databases for large sets of data
- crawl on simulated data set and actual crawl
Summary

- component framework for general tasks
- loosely-coupled field-specific components
- open, extensible, scalable architecture
- transparent caching mechanisms
- declarative configurability

Available for download at SourceForge.net

http://sourceforge.net/projects/webcrawler
Thank you for your attention!