



# LAW & the Web



---

Paolo Boldi, Massimo Santini e Sebastiano Vigna

# To Make a Table...

Per fare un tavolo ci vuole il legno  
per fare il legno ci vuole l'albero  
per fare l'albero ci vuole il seme  
per fare il seme ci vuole il frutto  
per fare il frutto ci vuole un fiore  
ci vuole un fiore, ci vuole un fiore,  
per fare un tavolo ci vuole un fio-o-re.

To make a table you need wood,  
To make wood, you need a tree.  
To make a tree, you need a seed,  
To make a seed, you need a fruit,  
To make a fruit, you need a flower.  
You need a flower, you need a flower,  
To make a table you need a flo-o-wer.

# How Everything Started

# How Everything Started

- ✓ About seven years ago, we (the *three musketeers*) were starting to be interested in the **structure of the Web** (as a graph, mainly)

# How Everything Started

- ✓ About seven years ago, we (the *three musketeers*) were starting to be interested in the **structure of the Web** (as a graph, mainly)
- ✓ We needed data (Web pages)

# How Everything Started

- ✓ About seven years ago, we (the *three musketeers*) were starting to be interested in the **structure of the Web** (as a graph, mainly)
- ✓ We needed data (Web pages)
- ✓ Large datasets were (are) difficult to get, because...

# How Everything Started

- ✓ About seven years ago, we (the *three musketeers*) were starting to be interested in the **structure of the Web** (as a graph, mainly)
- ✓ We needed data (Web pages)
- ✓ Large datasets were (are) difficult to get, because...
  - ✓ they are **large**

# How Everything Started

- ✓ About seven years ago, we (the *three musketeers*) were starting to be interested in the **structure of the Web** (as a graph, mainly)
- ✓ We needed data (Web pages)
- ✓ Large datasets were (are) difficult to get, because...
  - ✓ they are **large**
  - ✓ they are **precious**



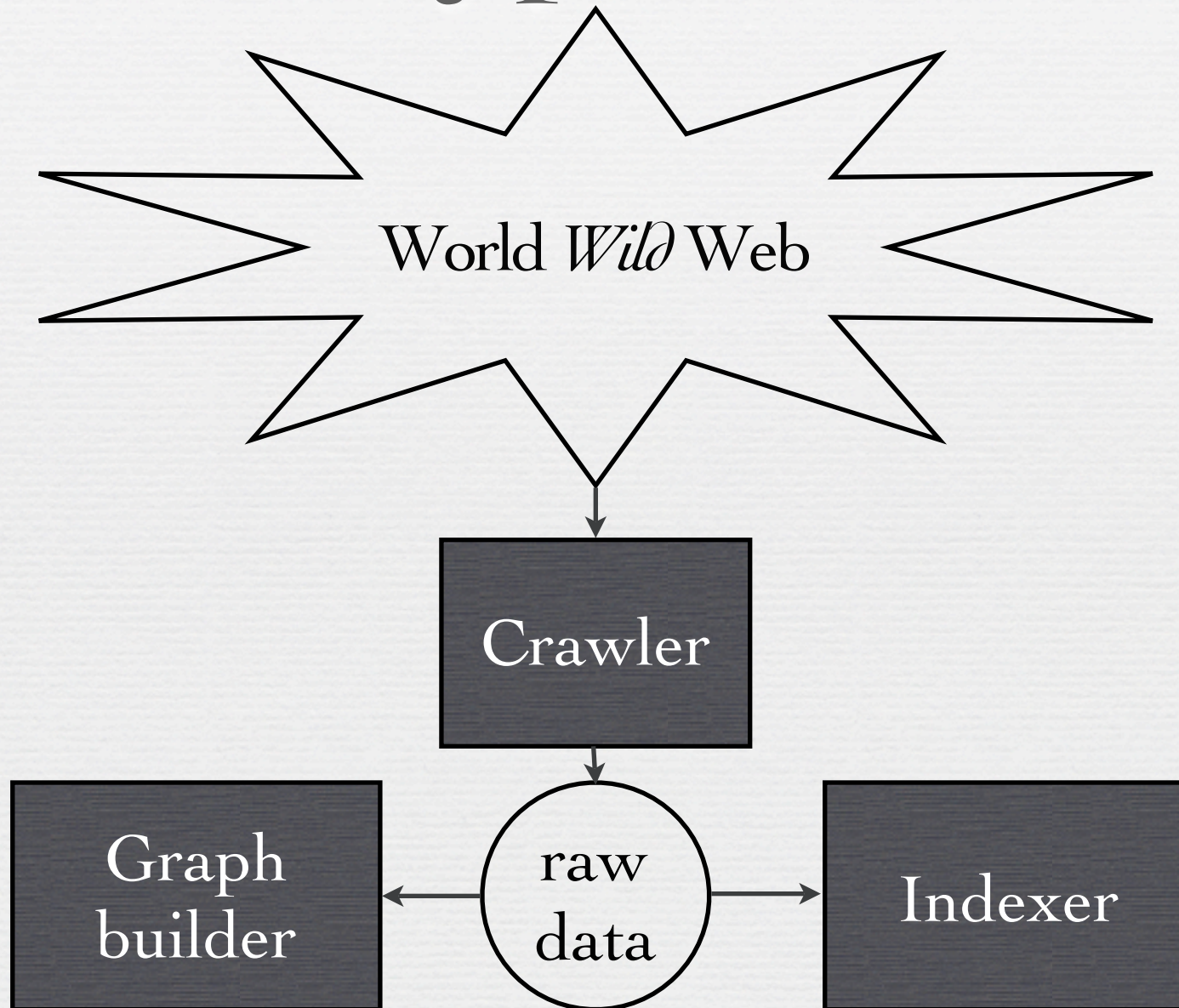


# How Everything Started

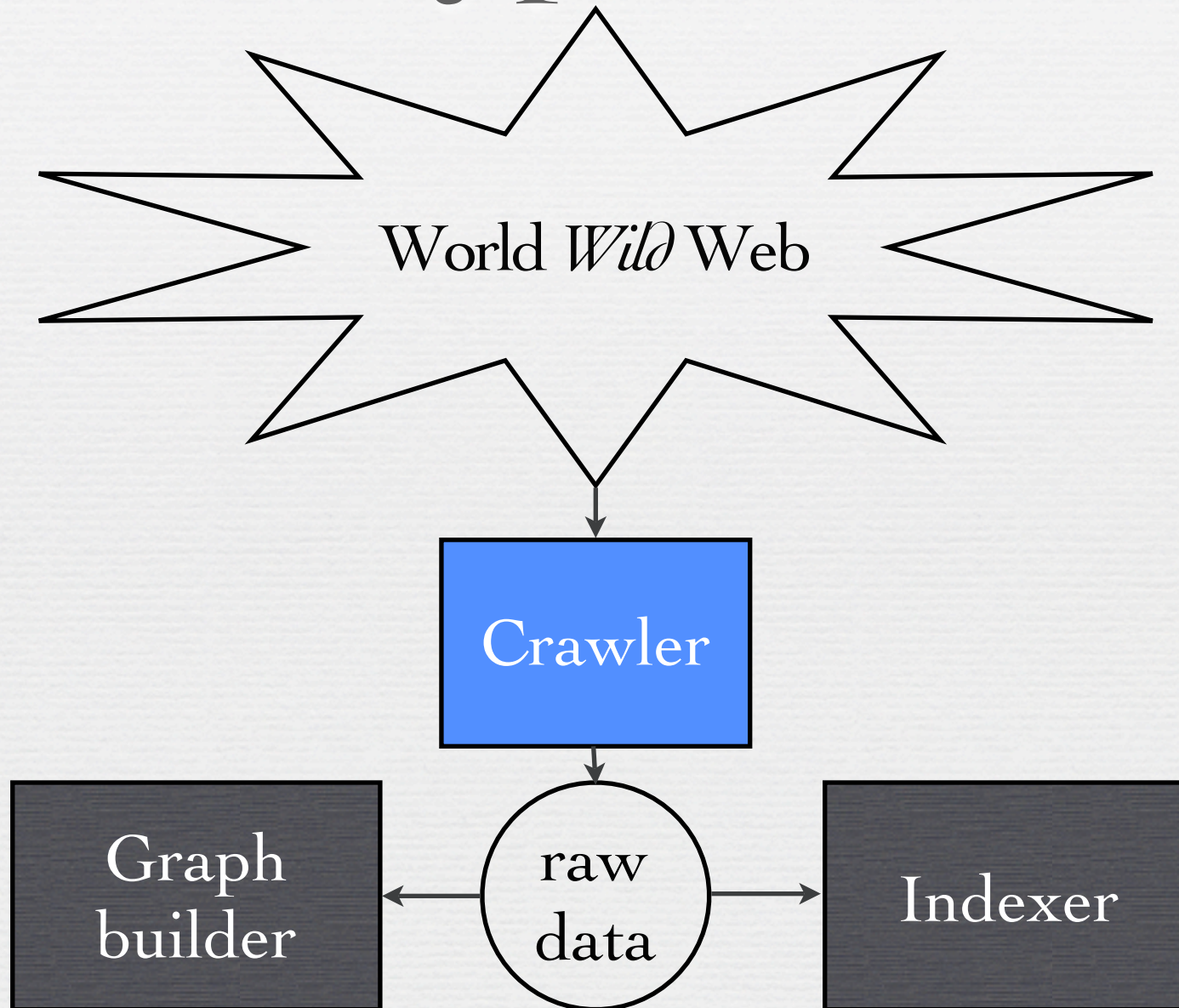
- ✓ About seven years ago, we (the *three musketeers*) were starting to be interested in the **structure of the Web** (as a graph, mainly)
- ✓ We needed data (Web pages)
- ✓ Large datasets were (are) difficult to get, because...
  - ✓ they are **large**
  - ✓ they are **precious**
  - ✓ they can be collected in many **different ways**



# (Proto) Typical Dataflow



# (Proto) Typical Dataflow



[B]Ubi[NG]

# UbiCrawler

# UbiCrawler

✓ Ubicumque + Crawler = UbiCrawler

# UbiCrawler

- ✓ **Ubicumque + Crawler = UbiCrawler**
- ✓ A 100% Java crawler

# UbiCrawler

- ✓ **Ubicumque + Crawler = UbiCrawler**
- ✓ A 100% Java crawler
- ✓ Was used to download more than 2,5 billion pages in the last four years (with 8 PCs, working erratically)





# UbiCrawler

- ✓ **Ubicumque + Crawler = UbiCrawler**
- ✓ A 100% Java crawler
- ✓ Was used to download more than 2,5 billion pages in the last four years (with 8 PCs, working erratically)
- ✓ Based on our experience with UbiCrawler, we are going to release a new, open-source crawler, called **BUbiNG** (Better Ubi / New Generation)



# UbiCrawler

- ✓ **Ubicumque + Crawler = UbiCrawler**
- ✓ A 100% Java crawler
- ✓ Was used to download more than 2,5 billion pages in the last four years (with 8 PCs, working erratically)
- ✓ Based on our experience with UbiCrawler, we are going to release a new, open-source crawler, called **BUbiNG** (Better Ubi / New Generation)



# BUbiNG Features

# BUbiNG Features

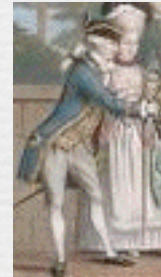
- ✓ Fully **distributed** (no central coordination)

# BUbiNG Features

- ✓ Fully **distributed** (no central coordination)
- ✓ **Scales up** almost linearly (more resources = better crawling)

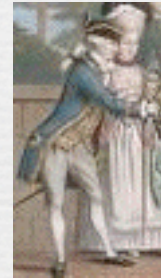
# BUbiNG Features

- ✓ Fully **distributed** (no central coordination)
- ✓ **Scales up** almost linearly (more resources = better crawling)
- ✓ **Polite** (more than UbiCrawler used to be...)



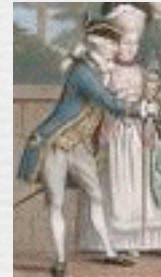
# BUbiNG Features

- ✓ Fully **distributed** (no central coordination)
- ✓ **Scales up** almost linearly (more resources = better crawling)
- ✓ **Polite** (more than UbiCrawler used to be...)
- ✓ Uses more **standard format** for saving data (Internet Archive's WARC, Draft ISO TC46/SC4)



# BUbiNG Features

- ✓ Fully **distributed** (no central coordination)
- ✓ **Scales up** almost linearly (more resources = better crawling)
- ✓ **Polite** (more than UbiCrawler used to be...)
- ✓ Uses more **standard format** for saving data (Internet Archive's WARC, Draft ISO TC46/SC4)
- ✓ Some more features in the following slides





# Consistent Hashing

# Consistent Hashing

- ✓ Identifier-seeded consistent hashing: an innovative way to distribute URLs among agents

# Consistent Hashing

- ✓ Identifier-seeded consistent hashing: an innovative way to distribute URLs among agents
- ✓ Locally computable: (pseudo)random function of agent names



# Consistent Hashing

- ✓ Identifier-seeded consistent hashing: an innovative way to distribute URLs among agents
- ✓ Locally computable: (pseudo)random function of agent names
- ✓ Somewhat between *static* and *dynamic* assignment:



# Consistent Hashing

- ✓ Identifier-seeded consistent hashing: an innovative way to distribute URLs among agents
- ✓ Locally computable: (pseudo)random function of agent names
- ✓ Somewhat between *static* and *dynamic* assignment:
  - ✓ as **new agents** are added, they become responsible of some URLs...



# Consistent Hashing

- ✓ Identifier-seeded consistent hashing: an innovative way to distribute URLs among agents
- ✓ Locally computable: (pseudo)random function of agent names
- ✓ Somewhat between *static* and *dynamic* assignment:
  - ✓ as **new agents** are added, they become responsible of some URLs...
  - ✓ ...but, except for this, they don't change the relative responsibilities of **existing agents**



JAI4J

# JAI4J

✓ **Job Assignment Infrastructure For Java**





# JAI4J



- ✓ **Job Assignment Infrastructure For Java**
- ✓ A general Java library to assign **jobs** (a.k.a. work unit) to **agents**: applicable to BUbiNG, Heritrix, ...

# JAI4J



- ✓ **J**ob **A**ssignment **I**nfrastructure **F**or **J**ava
- ✓ A general Java library to assign **jobs** (a.k.a. work unit) to **agents**: applicable to BUbiNG, Heritrix, ...
- ✓ Based on **JGroups** for communication/discovery etc.

# JAI4J



- ✓ **Job Assignment Infrastructure For Java**
- ✓ A general Java library to assign **jobs** (a.k.a. work unit) to **agents**: applicable to BUbiNG, Heritrix, ...
- ✓ Based on **JGroups** for communication/discovery etc.
- ✓ Uses **Berkeley DB** for keeping track of pending communications

# JAI4J



- ✓ **Job Assignment Infrastructure For Java**
- ✓ A general Java library to assign **jobs** (a.k.a. work unit) to **agents**: applicable to BUbiNG, Heritrix, ...
- ✓ Based on **JGroups** for communication/discovery etc.
- ✓ Uses **Berkeley DB** for keeping track of pending communications
- ✓ Will be released under LGPL (as part of BUbiNG)

**fastutil**

fastutil



# fastutil



✓ Java is more efficient than one often is lead to believe

# fastutil

- ✓ Java is more efficient than one often is lead to believe
- ✓ Yet, primitive-type efficient collections are still missing from the core APIs (and essential for any large-scale activity)



# fastutil

- ✓ Java is more efficient than one often is lead to believe
- ✓ Yet, primitive-type efficient collections are still missing from the core APIs (and essential for any large-scale activity)
- ✓ `fastutil` provides small-footprint, highly efficient Java implementations for primitive-type collections

# fastutil

- ✓ Java is more efficient than one often is lead to believe
- ✓ Yet, primitive-type efficient collections are still missing from the core APIs (and essential for any large-scale activity)
- ✓ `fastutil` provides small-footprint, highly efficient Java implementations for primitive-type collections
- ✓ More than 1,200 automatically generated classes

# fastutil

- ✓ Java is more efficient than one often is lead to believe
- ✓ Yet, primitive-type efficient collections are still missing from the core APIs (and essential for any large-scale activity)
- ✓ `fastutil` provides small-footprint, highly efficient Java implementations for primitive-type collections
- ✓ More than 1,200 automatically generated classes
- ✓ Released under LGPL

# Go Grab It!

## :::fastutil: Fast & compact type-specific collections for Java™

Home	
Publications	→
ERW	↔
<b>fastutil</b>	
Installation	
History and Motivation	
MG4J	↔
WebGraph	↔
ARSCIF	↔
Music	↔
Kitsch	↔

### Introduction

`fastutil` extends the [Java™ Collections Framework](#) by providing type-specific maps, sets, lists and queues with a small memory footprint and fast access and insertion; it also includes a fast I/O API for binary and text files. It is [free software](#) distributed under the [GNU Lesser General Public License](#).



The classes implement their standard counterpart interface (e.g., `Map` for maps) and can be plugged into existing code. Moreover, they provide additional features (such as bidirectional iterators) that are not available in the standard classes.

Besides objects and primitive types, `fastutil` classes provide support for *references*, that is, objects that are compared using the equality operator rather than the `equals()` method.

The sources are generated using a C preprocessor, starting from a set of driver files. You can peek at the javadoc-generated [documentation](#). In particular, the overview explains the design choices used in `fastutil`.

### Java™ 5

The release 5 of `fastutil` needs Java 5, and marks a revolution in the way the classes are organized. Besides full support for generics, which can help with object-based classes (but is rather irrelevant in type-specific classes), `fastutil` is now strongly based on what is probably the less hyped and most important feature of Java 5: *covariant return-type overriding*. A method `x()` returning an object of type  $\tau$  can now be overridden by a method returning an object of type  $\upsilon$ , where  $\upsilon$  is a subclass of (or implements)  $\tau$ .

Covariant return-type overriding has always been supported by the JVM, but was inaccessible at the syntax level. In Java 5 this irritant limitation has been lifted, opening a new world of clarity and orthogonality in organizing collection classes. In particular, *interface strengthening* makes it possible to use all features of `fastutil` classes without type casting: for instance, `IntSet.iterator()` is strengthened, w.r.t. `Set.iterator()`, so that it returns an `IntIterator` instead of a simple `Iterator`. Although type casting was previously used to simulate this feature, it made using the hundreds of classes in `fastutil` difficult and error-prone, as every type cast had to be checked against the documentation.

Google™

Search

Download

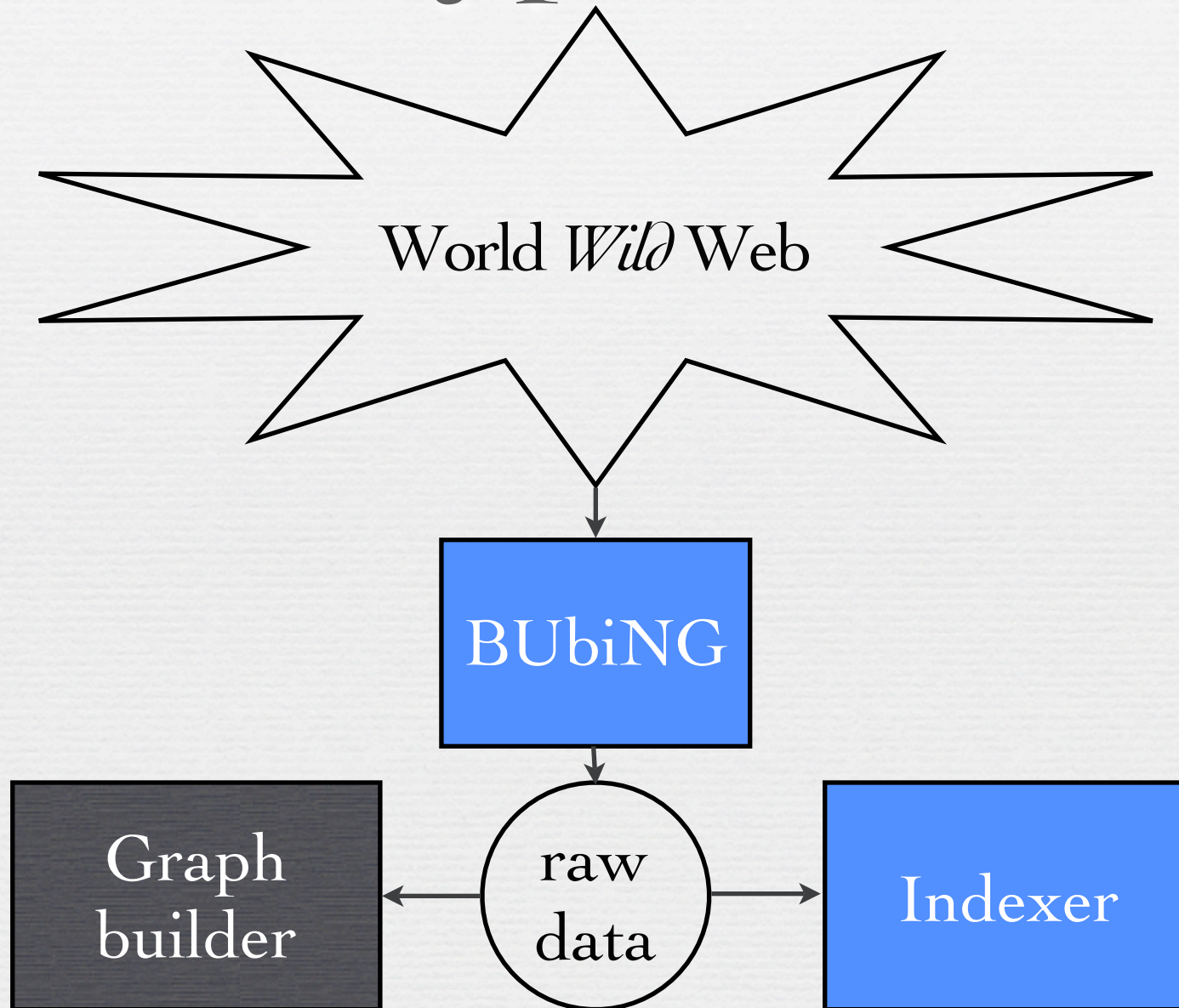
- [binary tarball \(Java ≥5\)](#)
- [source tarball \(Java ≥5\)](#)
- [binary tarball \(unmaintained older version, Java <5\)](#)
- [source tarball \(unmaintained older version, Java <5\)](#)
- [JPackage RPMs](#)

Documentation

- [README](#)
- [CHANGES](#)
- [API \(Java ≥5\)](#)
- [API \(unmaintained, Java <5\)](#)

This is valid HTML  
4.01

# (Proto) Typical Dataflow



MG4J

MG4J

# MG4J

✓ Managing Gigabytes For Java





# MG4J



- ✓ **Managing Gigabytes For Java**
- ✓ Started as a Java partial implementation of world-famous MG (Managing Gigabytes)

# MG4J



- ✓ **Managing Gigabytes For Java**
- ✓ Started as a Java partial implementation of world-famous MG (Managing Gigabytes)
- ✓ Now: a full-fledged set of facilities to index and full-text search large document collections

# MG4J



- ✓ **Managing Gigabytes For Java**
- ✓ Started as a Java partial implementation of world-famous MG (Managing Gigabytes)
- ✓ Now: a full-fledged set of facilities to index and full-text search large document collections
- ✓ On the way, besides optimized implementations, we developed several new algorithms

# MG4J Features

# MG4J Features

- ✓ Highly customizable: document collections can be potentially anything, flexible index structure (ideal for research purposes)

# MG4J Features

- ✓ Highly customizable: document collections can be potentially anything, flexible index structure (ideal for research purposes)
- ✓ Extremely efficient; suitable for **large datasets**



# MG4J Features

- ✓ Highly customizable: document collections can be potentially anything, flexible index structure (ideal for research purposes)
- ✓ Extremely efficient; suitable for **large datasets**
- ✓ Multi-field search, virtual fields, distributed search



# MG4J Features

- ✓ Highly customizable: document collections can be potentially anything, flexible index structure (ideal for research purposes)
- ✓ Extremely efficient; suitable for **large datasets**
- ✓ Multi-field search, virtual fields, distributed search
- ✓ Includes low-level Java libraries for





# MG4J Features

- ✓ Highly customizable: document collections can be potentially anything, flexible index structure (ideal for research purposes)
- ✓ Extremely efficient; suitable for **large datasets**
- ✓ Multi-field search, virtual fields, distributed search
- ✓ Includes low-level Java libraries for
  - ✓ highly efficient mutable strings



# MG4J Features

- ✓ Highly customizable: document collections can be potentially anything, flexible index structure (ideal for research purposes)
- ✓ Extremely efficient; suitable for **large datasets**
- ✓ Multi-field search, virtual fields, distributed search
- ✓ Includes low-level Java libraries for
  - ✓ highly efficient mutable strings
  - ✓ bit-level I/O and compression



# MG4J Features

- ✓ Highly customizable: document collections can be potentially anything, flexible index structure (ideal for research purposes)
- ✓ Extremely efficient; suitable for **large datasets**
- ✓ Multi-field search, virtual fields, distributed search
- ✓ Includes low-level Java libraries for
  - ✓ highly efficient mutable strings
  - ✓ bit-level I/O and compression
  - ✓ (OP)MPH, FCL, etc.



# Minimal-Interval Semantics

# Minimal-Interval Semantics

- ✓ MG4J full-text search adopts **minimal-interval semantics**: if a document satisfies a query, there is a set of minimal text intervals that witness it

# Minimal-Interval Semantics

- ✓ MG4J full-text search adopts **minimal-interval semantics**: if a document satisfies a query, there is a set of minimal text intervals that witness it
- ✓ All query operators are translated into operators between minimal intervals

# Minimal-Interval Semantics

- ✓ MG4J full-text search adopts **minimal-interval semantics**: if a document satisfies a query, there is a set of minimal text intervals that witness it
- ✓ All query operators are translated into operators between minimal intervals
- ✓ The latter have new almost **linear** and **optimally lazy** implementations: never advanced unless it is actually required!

# Minimal-Interval Semantics

- ✓ MG4J full-text search adopts **minimal-interval semantics**: if a document satisfies a query, there is a set of minimal text intervals that witness it
- ✓ All query operators are translated into operators between minimal intervals
- ✓ The latter have new almost **linear** and **optimally lazy** implementations: never advanced unless it is actually required!





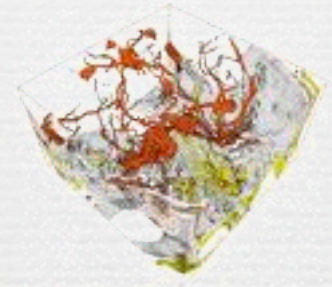
# Minimal-Interval Semantics

# Minimal-Interval Semantics

- ✓ Minimal-interval semantics + multiple fields = very rich and **powerful query language**

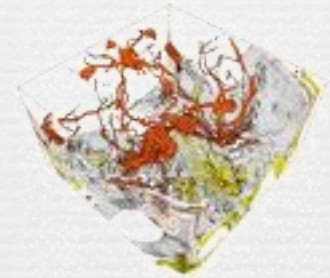
# Minimal-Interval Semantics

- ✓ Minimal-interval semantics + multiple fields = very rich and **powerful query language**
- ✓ Also: a complex and sophisticated semantics



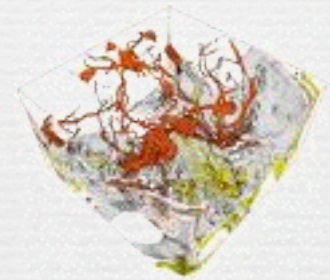
# Minimal-Interval Semantics

- ✓ Minimal-interval semantics + multiple fields = very rich and **powerful query language**
- ✓ Also: a complex and sophisticated semantics
- ✓ Needs correspondingly complex algebraic tools to be described (free product [a.k.a. categorical sum] of lattices)



# Minimal-Interval Semantics

- ✓ Minimal-interval semantics + multiple fields = very rich and **powerful query language**
- ✓ Also: a complex and sophisticated semantics
- ✓ Needs correspondingly complex algebraic tools to be described (free product [a.k.a. categorical sum] of lattices)
- ✓ Full description is coming...



# Skip Lists

# Skip Lists

- ✓ *MG4J* uses deterministic, embedded, and highly compressed skip lists to speed up index access

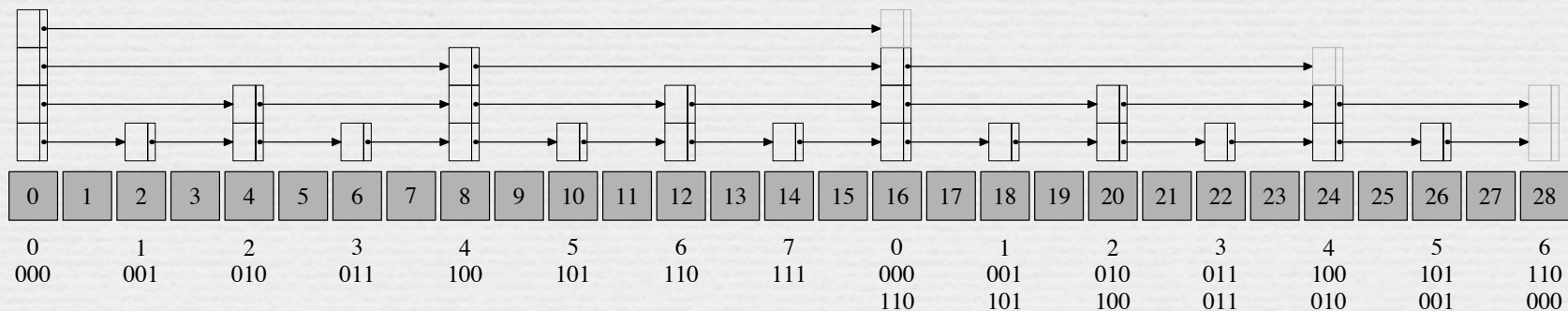
# Skip Lists

- ✓ MG4J uses deterministic, embedded, and highly compressed skip lists to speed up index access
- ✓ The classical approach (one skip every  $f^{1/2}$  pointers,  $f$  being the frequency) is tied to term-at-a-time query resolution in non-positional indices



# Skip Lists

- ✓ MG4J uses deterministic, embedded, and highly compressed skip lists to speed up index access
- ✓ The classical approach (one skip every  $f^{1/2}$  pointers,  $f$  being the frequency) is tied to term-at-a-time query resolution in non-positional indices
- ✓ MG4J's approach is agnostic (works even for sampling) and guarantees logarithmic access times with minimal space overhead




# Go Grab It!

## :::MG4J: Managing Gigabytes for Java™

Home	↔
Publications	↔
Software	↔
ERW	↔
fastutil	↔
<b>MG4J</b>	↔
Why Java?	↔
Installation	↔
WebGraph	↔
ARSCIF	↔
Music	↔
Kitsch	↔

### Introduction

MG4J (*Managing Gigabytes for Java*) is a free full-text search engine for large document collections written in Java. As a by-product, it offers several general-purpose optimised classes, including [fast & compact mutable strings](#), [bit-level I/O](#), (possibly signed) [minimal perfect hashing for very large strings collections](#), etc. 

With release 1.1, MG4J becomes a highly customisable, high-performance, full-fledged search engine providing state-of-the-art features (such as [BM25 scoring](#)) and new research algorithms.

The main points of MG4J are:

- **Powerful indexing.** Support for document collections and factories makes it possible to analyse, index and query consistently large document collections, providing easy-to-understand snippets that highlight relevant passages in the retrieved documents.
- **Efficiency.** We do not provide meaningless data such as "we index x GiB per second" (with which configuration? which language? which data source?)—we invite you to try it. MG4J can index without effort the TREC GOV2 collection (document factories are provided to this purpose) and scales to hundreds of millions of documents.
- **Multi-index interval semantics.** When you submit a query, MG4J returns, for each index, a list of intervals satisfying the query. This provides the base for several high-precision scorers and for very efficient implementation of sophisticated operators. The intervals are built in linear time using [new research algorithms](#).
- **Expressive operators.** MG4J goes far beyond the bag-of-words model, providing efficient implementation of phrase queries, proximity restrictions, ordered conjunction, and combined multiple-index queries. Each operator is represented internally by an abstract object, so you can easily plug in your favourite syntax.
- **Virtual fields.** MG4J supports *virtual fields*—fields containing text for a different, *virtual document*, the typical example is anchor text, which must be attributed to the target document.

Google™

Search

### Download

- [binary tarball](#)
- [source tarball](#)
- [dependencies tarball](#)
- [JPackage RPMs](#)

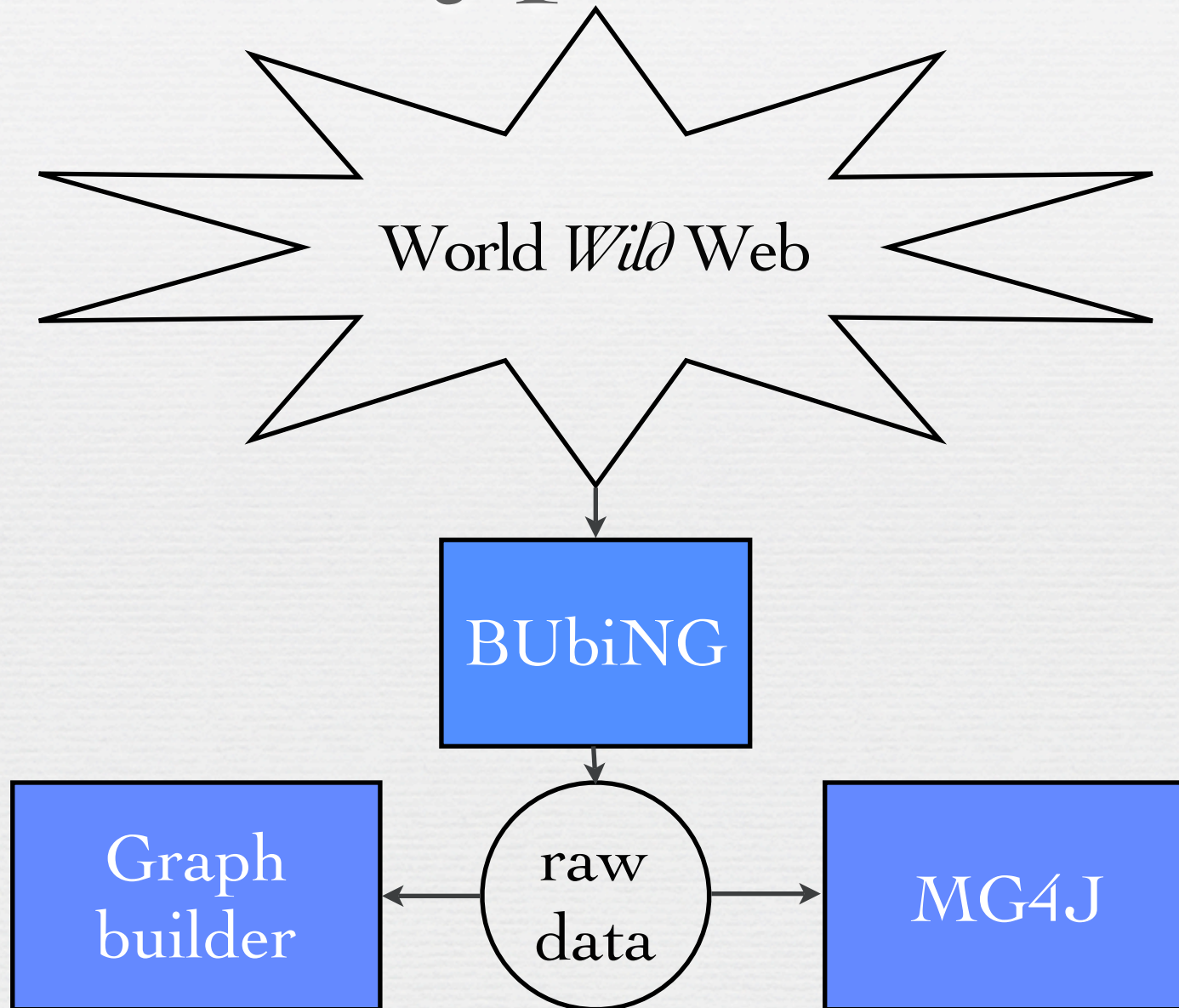
### Documentation

- [README](#)
- [CHANGES](#)
- [Manual \(HTML\)](#)
- [Manual \(PDF\)](#)
- [API](#)

### Papers

- A [paper](#) about the high-performance reimplementation of strings provided by the versatile MG4J class [MutableString](#), and [compact approximators](#), the randomised data structure used in [TextPattern](#) to represent bad-character shifts.
- A [preprint](#) about the skipping structures used in MG4J (a short version appeared in the proceedings of [SPIRE 2005](#)).
- A [preprint](#) about the algorithms for minimal-interval semantics used in MG4J (a shorter version appeared in the proceedings of [SPIRE 2005](#)).

# (Proto) Typical Dataflow



# WebGraph

# Extracting the Graph

# Extracting the Graph

- ✓ The **Web graph** contains a lot of information about the dataset



# Extracting the Graph

- ✓ The **Web graph** contains a lot of information about the dataset
- ✓ Many applications:



# Extracting the Graph

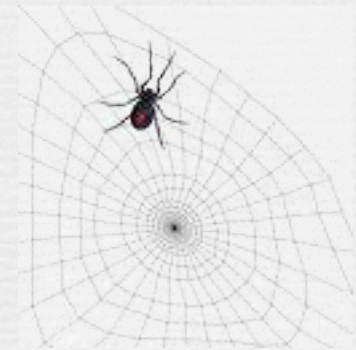
- ✓ The **Web graph** contains a lot of information about the dataset
- ✓ Many applications:
  - ✓ ranking





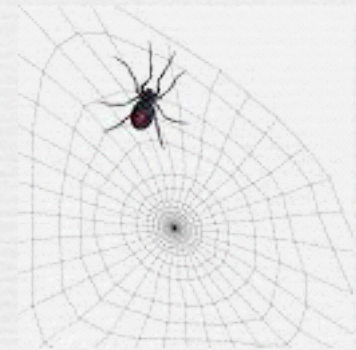
# Extracting the Graph

- ✓ The **Web graph** contains a lot of information about the dataset
- ✓ Many applications:
  - ✓ ranking
  - ✓ community detection



# Extracting the Graph

- ✓ The **Web graph** contains a lot of information about the dataset
- ✓ Many applications:
  - ✓ ranking
  - ✓ community detection
  - ✓ structural similarity



# WebGraph

# WebGraph


- ✓ Web graphs can be handled more easily than an entire crawl, provided that they are **suitably compressed**

# WebGraph


- ✓ Web graphs can be handled more easily than an entire crawl, provided that they are **suitably compressed**




# WebGraph

- ✓ Web graphs can be handled more easily than an entire crawl, provided that they are **suitably compressed** 
- ✓ The **WebGraph framework** provides out-of-the-box *state-of-the-art* compression requiring 3 bits/link on the average at 150ns/link random access time

# WebGraph


- ✓ Web graphs can be handled more easily than an entire crawl, provided that they are **suitably compressed** 
- ✓ The **WebGraph framework** provides out-of-the-box *state-of-the-art* compression requiring 3 bits/link on the average at 150ns/link random access time
- ✓ Medium-sized crawl graphs (e.g., 250Mpages) can be fit into the main memory of a workstation (4GB)

# WebGraph

- ✓ Web graphs can be handled more easily than an entire crawl, provided that they are **suitably compressed** 
- ✓ The **WebGraph framework** provides out-of-the-box *state-of-the-art* compression requiring 3 bits/link on the average at 150ns/link random access time
- ✓ Medium-sized crawl graphs (e.g., 250Mpages) can be fit into the main memory of a workstation (4GB)
- ✓ 100% Java, distributed under GPL



# WebGraph

- ✓ Web graphs can be handled more easily than an entire crawl, provided that they are **suitably compressed** 
- ✓ The **WebGraph framework** provides out-of-the-box *state-of-the-art* compression requiring 3 bits/link on the average at 150ns/link random access time
- ✓ Medium-sized crawl graphs (e.g., 250Mpages) can be fit into the main memory of a workstation (4GB)
- ✓ 100% Java, distributed under GPL
- ✓ Jacob Ratkiewicz has developed a C++ version

# Compression

# Compression

✓ WebGraph uses a mixture of:

# Compression

- ✓ WebGraph uses a mixture of:
  - ✓ **locality** (similar URLs point one another)

# Compression

- ✓ WebGraph uses a mixture of:
  - ✓ **locality** (similar URLs point one another)
  - ✓ **similarity** (similar URLs have many links in common)

# Compression

- ✓ WebGraph uses a mixture of:
  - ✓ **locality** (similar URLs point one another)
  - ✓ **similarity** (similar URLs have many links in common)
  - ✓ **consecutivity** (similarity in the transposed graph)

# Compression

- ✓ WebGraph uses a mixture of:
  - ✓ **locality** (similar URLs point one another)
  - ✓ **similarity** (similar URLs have many links in common)
  - ✓ **consecutivity** (similarity in the transposed graph)
  - ✓ new, made-to-measure **universal codes**

# $\zeta$ Codes



# $\zeta$ Codes

- ✓ Existing universal codes (e.g., Golomb, Elias' gamma, etc.) often target distributions that are *not* found in the Web

# $\zeta$ Codes

- ✓ Existing universal codes (e.g., Golomb, Elias' gamma, etc.) often target distributions that are *not* found in the Web
- ✓ We developed a new family of codes ( **$\zeta$  codes**) that are intended for Power Law distributions with low exponent ( $<2$ )



# $\zeta$ Codes

- ✓ Existing universal codes (e.g., Golomb, Elias' gamma, etc.) often target distributions that are *not* found in the Web
- ✓ We developed a new family of codes ( **$\zeta$  codes**) that are intended for Power Law distributions with low exponent ( $<2$ )



$\alpha$	Huffman code			$\zeta$ codes			
	Entropy	Exp. length	Red.	$H_\alpha$	Code	Exp. length	Red.
1.15	11.37	11.39	0.23%	13.01	$\zeta_4$	13.61	4.66%
1.20	9.62	9.66	0.46%	10.20	$\zeta_3$	10.57	3.56%
1.25	8.24	8.27	0.37%	8.45	$\zeta_3$	8.72	3.21%
1.30	7.16	7.19	0.28%	7.24	$\zeta_2$	7.44	2.69%
1.35	6.32	6.34	0.36%	6.35	$\zeta_2$	6.47	1.86%
1.40	5.64	5.68	0.63%	5.65	$\zeta_2$	5.75	1.65%
1.45	5.10	5.16	1.18%	5.10	$\zeta_2$	5.20	1.95%
1.50	4.64	4.70	1.16%	4.64	$\zeta_2$	4.77	2.68%
1.55	4.26	4.29	0.82%	4.26	$\zeta_2$	4.42	3.79%
1.60	3.93	3.95	0.60%	3.93	$\zeta_1$	4.07	3.61%


# Go Grab It!

## :::WebGraph

- Home
- Publications
- Software
- ERW
- fastutil
- MG4J
- WebGraph**
- Installation
- WebGraph++
- Datasets
- ARSCIF
- Music
- Kitsch


### Introduction


WebGraph is a framework to study the web graph. It provides simple ways to manage very large graphs, exploiting modern compression techniques. More precisely, it is currently made of:



1. A set of flat codes, called  $\zeta$  codes, which are particularly suitable for storing web graphs (or, in general, integers with power-law distribution in a certain exponent range). The fact that these codes work well can be easily tested empirically, but we also try to provide a [detailed mathematical analysis](#).
2. Algorithms for compressing web graphs that exploit gap compression and referentiation (à la [LINK](#)), intervalisation and  $\zeta$  codes to provide a high compression ratio: for instance, the [WebBase](#) graph (2001 crawl) is compressed at 3.08 bits per link, and a snapshot of about 18,500,000 pages of the .uk domain gathered by [UbiCrawler](#) is compressed at 2.22 bits per link (the corresponding figures for the transposed graphs are 2.89 bits per link and 1.98 bits per link). The algorithms are controlled by several parameters, which provide different tradeoffs between access speed and compression ratio.
3. Algorithms for accessing a compressed graph without actually decompressing it, using lazy techniques that delay the decompression until it is actually necessary.
4. A complete, [documented](#) implementation of the algorithms above in Java, contained in the package `it.unimi.dsi.webgraph`. Besides a clearly defined API, the package contains several classes that allow to modify (e.g., transpose) or recompress a graph, so to experiment with various settings. The package relies on [fastutil](#) 5 for a type-specific, high-performance collections framework, on [MG4J](#) for bit-level I/O, on the [COLT distribution](#) for ready-to-use, efficient algorithms and on [JSAP](#) for line-command parsing.
5. [Datasets](#) for very large graph (e.g., a billion of links). These are either gathered from public sources (such as [WebBase](#)), or produced by [UbiCrawler](#).

In the end, with WebGraph you can access and analyse a very large web graph, even on a PC with as little as 256 Mbytes of RAM. Using WebGraph is as easy as installing a few jar files and downloading a dataset. This makes studying phenomena such as PageRank, distribution of graph properties of the web graph, etc





Download (Java  $\geq$ 5)

- [binary tarball](#)
- [source tarball](#)
- [dependencies tarball](#)
- [JPackage-like RPM](#)
- [JPackage-like Javadoc RPM](#)
- [JPackage-like source RPM](#)
- [Datasets](#)

Documentation

- [API](#)
- [CHANGES](#)

Papers

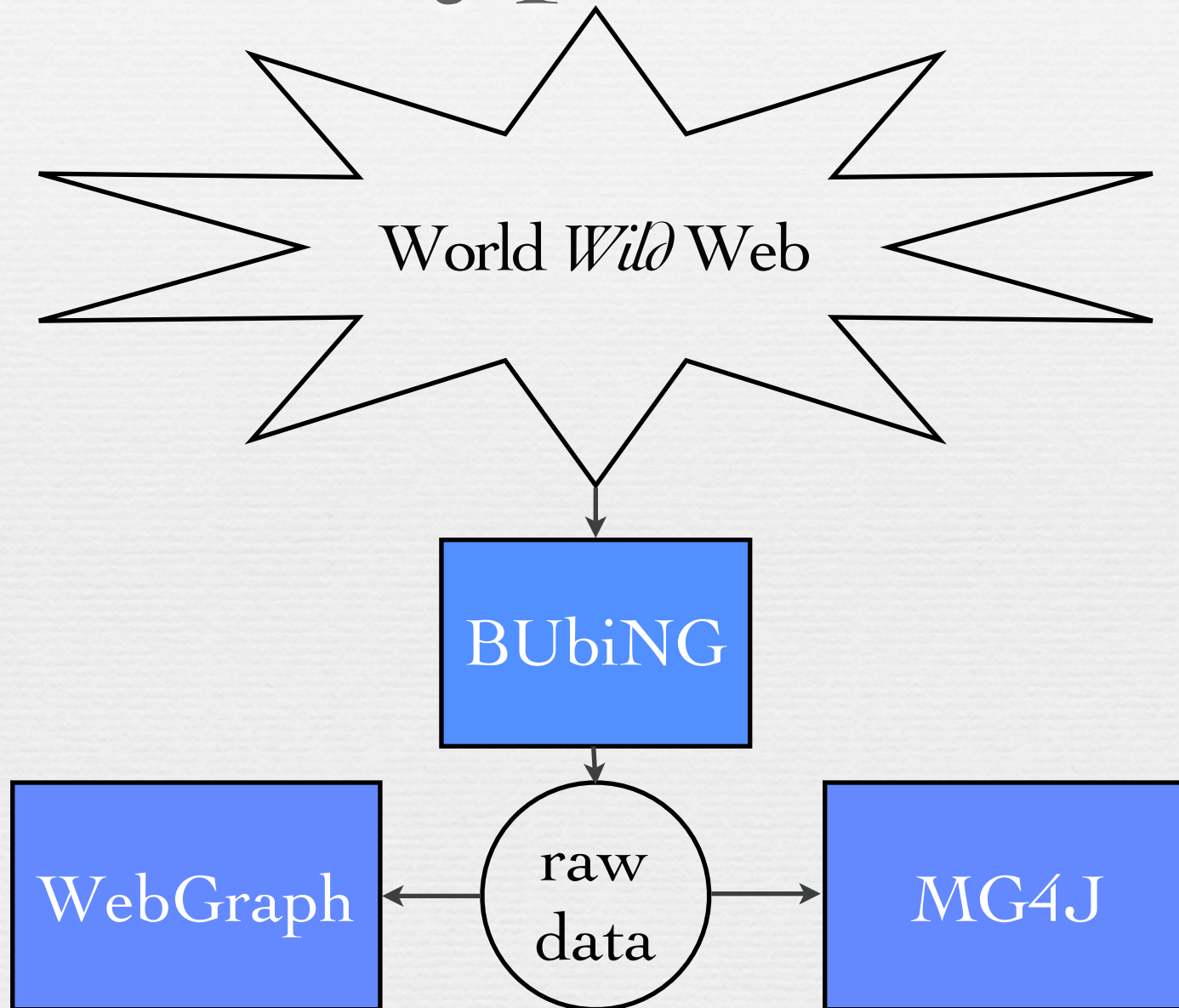
- A [detailed description](#) of the compression algorithms used in WebGraph, published in the proceedings of the [Thirteenth International World-Wide Web Conference](#).
- A [mathematical analysis](#) of the performance of  $\gamma$ ,  $\delta$  and  $\zeta$  codes against power-law distributions.

Trivia

When WebGraph was released, it was featured on [Slashdot](#).

This is valid HTML 4.01

# (Proto) Typical Dataflow



# LAW Library

# PageRank and Derivatives



# PageRank and Derivatives



- ✓ The LAW library contains software to



# PageRank and Derivatives



- ✓ The LAW library contains software to
  - ✓ compute **PageRank** in various flavors (using the Power Method, Gauss-Seidel, Jacobi etc.)

# PageRank and Derivatives



- ✓ The LAW library contains software to
  - ✓ compute **PageRank** in various flavors (using the Power Method, Gauss-Seidel, Jacobi etc.)
  - ✓ compute its **Power Series Expansion** and evaluate it simultaneously for *different damping factors* and for *derivatives* of any order with precision guarantees


# PageRank and Derivatives



- ✓ The LAW library contains software to
  - ✓ compute **PageRank** in various flavors (using the Power Method, Gauss-Seidel, Jacobi etc.)
  - ✓ compute its **Power Series Expansion** and evaluate it simultaneously for *different damping factors* and for *derivatives* of any order with precision guarantees
  - ✓ compare (efficiently) two ranks using **Kendall's  $\tau$**

# Go Grab It!

## Laboratory for Web Algorithmics



### Main Menu

- [Home](#)
- [Software](#)
- [Hardware](#)
- [Datasets](#)
- [Tesi](#)
- [Publications](#)
- [Collaborations](#)
- [Teaching](#)
- [People](#)

### Login Form


Username  
admin

Password  
\*\*\*\*\*

Remember me

[Password Reminder](#)  
No account yet? [Create one](#)


### Style



c7\_doopal

Select

### Syndicate

 0.91


## Introduction to LAW

The Laboratory for Web Algorithmics (LAW) was established in 2002 at the **Dipartimento di Scienze dell'Informazione (DSI)** of the **Università degli studi di Milano**.

Research at LAW concerns all algorithmic aspects of the web. More in detail:


### High-performance parallel web crawling

**UbiCrawler** is a scalable, fault-tolerant and fully distributed web crawler developed in collaboration with the **Istituto di Informatica e Telematica**. The **first report on the design of UbiCrawler** won the Best Poster Award at the **Tenth World Wide Web Conference**.



### Web-graph compression

Once a part of the web has been crawled, the resulting graph is very large—you need a compact representation. **WebGraph** is a framework built to this purpose. Among other things, WebGraph uses new instantaneous codes for the integers and new aggressive algorithmic compression techniques.



### Web-graph analysis

Web graphs have special properties whose study requires a sizeable amount of mathematics, but also a careful study of *actual* web graphs. We have studied, for instance, **the paradoxical way PageRank evolves during a crawl**, and **the way PageRank changes depending on the damping factor**.

### Search-engine construction

Often, the purpose of a crawl is the construction of a *full-text index* of the text contained in the crawled pages. Such an index is at the basis of all existing commercial search engines such as **Google**.

The research of search-engine construction is based on **MG4J**, a system for full-text indexing of

### Search

search...

### Newsflash

LAW released version 1.1 of its **software bundle**. Now PageRank can be computed with three different algorithms (power method, Gauss-Seidel, and Jacobi). Moreover all algorithms have in-depth explanations both from a mathematical and an algorithmic viewpoint. Consistent hashing computation (formerly in the UbiCrawler external package) has been included.

### Recent items

- **Compressione differenziale di file WARC**
- **LAW releases updated software**
- **New dataset section**
- **WebGraph 1.5**
- **New Technical Report**

### News

chance of victory in the first Gulf War was 93%, while the poor Soviets only had a 7% chance in Afghanistan (if only they'd known; failure maybe

# Finally, the Datasets

# The DELIS Dataset

# The DELIS Dataset

- ✓ Various datasets over the last 5 years

# The DELIS Dataset

- ✓ Various datasets over the last 5 years
- ✓ The most complete is the DELIS one



# The DELIS Dataset

- ✓ Various datasets over the last 5 years
- ✓ The most complete is the DELIS one
  - ✓ 12 snapshots of the UK web

# The DELIS Dataset

- ✓ Various datasets over the last 5 years
- ✓ The most complete is the DELIS one
  - ✓ 12 snapshots of the UK web
  - ✓ monthly, from the same seed

# The DELIS Dataset

- ✓ Various datasets over the last 5 years
- ✓ The most complete is the DELIS one
  - ✓ 12 snapshots of the UK web
  - ✓ monthly, from the same seed
  - ✓ 100Mpages each

# The DELIS Dataset

- ✓ Various datasets over the last 5 years
- ✓ The most complete is the DELIS one
  - ✓ 12 snapshots of the UK web
  - ✓ monthly, from the same seed
  - ✓ 100Mpages each
- ✓ Graphs available in *differential* form: a single, *aligned* WebGraph with labels on nodes/arcs —less than 7GB!

# Full Text

# Full Text

Date	Pages	Size (Gb)	GZip Size (Gb)
2006-05	98,812,333.00	2,014.68	419.57
2006-06	112,386,763.00	1,893.11	402.45
2006-07	136,956,559.00	2,287.36	477.03
2006-08	141,395,895.00	2,424.82	507.59
2006-09	148,965,298.00	2,756.61	546.70
2006-10	129,558,491.00	2,336.19	478.31
2006-11	150,146,132.00	2,637.70	546.81
2006-12	144,489,446.00	2,552.80	525.77
2007-01	151,578,113.00	2,651.65	553.97
2007-02	153,966,540.00	2,692.88	564.98
2007-03	151,427,461.00	2,568.80	545.80
2007-04	150,606,689.00	2,700.06	559.84

# Graphs

# Graphs

Dataset	Nodes	Arcs	Size (Gb)	bit/arc
2006-05	77,741,046	2,965,197,340	1.10	2.90
2006-06	80,644,902	2,481,281,617	0.99	3.08
2006-07	96,395,298	3,030,665,444	1.28	3.30
2006-08	100,751,978	3,250,153,746	1.35	3.26
2006-09	106,288,541	3,871,625,613	1.45	2.93
2006-10	93,463,772	3,130,910,405	1.14	2.83
2006-11	106,783,458	3,479,400,938	1.29	2.86
2006-12	103,098,631	3,768,836,665	1.34	2.78
2007-01	108,563,230	3,929,837,236	1.38	2.72
2007-02	110,123,614	3,944,932,566	1.39	2.74
2007-03	107,565,084	3,642,701,825	1.34	2.85
2007-04	106,867,191	3,790,305,474	1.36	2.79



Overlap (host)



# Go Grab It (someday...)!



## DSI and DELIS

The [DELIS](#) (Dynamically Evolving Large-scale Information Systems) European project deals with methods, techniques, tools, and prototypical implementations in order to cope with challenges imposed by the size and dynamics of today's and especially future information systems. Special emphasis is on exploiting synergies between the different groups that are part of the project. The fundamental features of the way networks and systems are studied within DELIS stem from the focus on how individual (user) node behavior impacts the network as a whole, and the focus on whether it is feasible and how to attain a useful networking infrastructure, given individual, self-centered behavior. Thus, the key technical challenges in DELIS aim to create a self-organizing, self-repairing, self-monitoring network infrastructure, given largely autonomous, perhaps egoistic and selfish peers, coexisting with altruists. The DELIS subproject structure exemplifies this.

The [DSI](#) (Computer Science Department) of the [Università di Milano](#) is part of the Subproject 1 (*Monitoring, Visualizing, and Analyzing Large Dynamically Evolving Information Systems*) aimed, in particular, at the analysis of the structure of large dynamic networks, specifically of the *web graph*: given a portion of the web, its web graph is a [directed graph](#) whose nodes are the web pages, and with an arc from node *A* to node *B* iff there is a hypertextual link to page *B* contained in the text of page *A*. Web graphs are of utmost importance in various fields such as, for example, in the way search engines organize their results, in the search for virtual communities, in the analysis of web spam etc.

With the aim of studying real-world web graph and their evolution in time, we started collecting in a systematic way large portions of the Web; this is the starting point to allow other partners, for example, to assess their statements and models about the temporal evolution of the statistical and topological properties of the web graph and the possible impact of different crawling strategies.

## How/when/where the snapshots were collected

Together with our DELIS partners, we decided to take snapshots at a monthly rate focussing on the .uk domain: the choice of the domain was the most obvious, given the European nature of the project and in consideration of the linguistic and social centrality of the UK within Europe; the frequency chosen is the largest possible that does not raise issues of unpoliteness. The first snapshot was collected in May 2006. All snapshots were collected at the [DSI](#), using [hardware](#) that was partly funded by the DELIS project.

All the data sets were collected using [UbiCrawler](#), a scalable, fault-tolerant and fully distributed web crawler developed by the [Laboratory for Web Algorithmics \(LAW\)](#) at the [DSI](#), and the corresponding web graphs are made available in this page in the [WebGraph](#) compressed format — please, read the [tutorial](#) to learn how to use such format.

## The snapshots

### Full-texts

Dataset	Pages	Size (Gb)	GZip Size (Gb)
uk-2006-05	98812333	2014.68	419.57
uk-2006-06	112386763	1893.11	402.45
uk-2006-07	136956559	2287.36	477.03
uk-2006-08	141395895	2424.82	507.59
uk-2006-09	148965298	2756.61	546.70
uk-2006-10	120558401	2226.10	478.21

# References

- ✓ BSV (with Bruno Codenotti). **Ubicrawler: A scalable fully distributed web crawler.** *Software: Practice & Experience*, 34(8):711-726, 2004.
- ✓ BV. **The WebGraph framework I: Compression techniques.** In Proc. WWW 2004, pages 595-601, Manhattan, USA, 2004. ACM Press.
- ✓ BV. **WebGraph: Things you thought you could not do with Java™.** In Proc. of the 3rd PPPJ, ACM International Conference Proceedings Series, pages 1-8, Las Vegas, Nevada, USA, 2004.
- ✓ BV. **Mutable strings in Java: Design, implementation and lightweight text-search algorithms.** *Sci. Comput. Programming*, 54(1):3-23, 2005.
- ✓ BV. **Efficient lazy algorithms for minimal-interval semantics.** In Proc. SPIRE 2006, LNCS 4209, pages 134-149. Springer-Verlag, 2006.
- ✓ BV. **Compressed perfect embedded skip lists for quick inverted-index lookups.** In Proc. SPIRE 2005, LNCS 3772, pages 25-28. Springer-Verlag, 2005.

# References

(more theoretical flavor)

- ✓ BSV. **Do your worst to make the best: Paradoxical effects in PageRank incremental computations.** *Internet Math.*, 2(3):387-404, 2005.
- ✓ BSV. **PageRank as a function of the damping factor.** In *Proc. WWW 2005*, pages 557-566, Chiba, Japan, 2005. ACM Press.
- ✓ BV. **Codes for the World-Wide Web.** *Internet Math.*, 2(4):405-427, 2005.
- ✓ BSV (with Roberto Posenato). **Traps and pitfalls of topic-biased PageRank.** In *WAW 2006, LNCS*. Springer-Verlag, 2007.



