

Crawling important sites on the Web

Serge Abiteboul, Mihai Preda, Grégory Cobéna

Research performed at INRIA and now commercialized by Xyleme

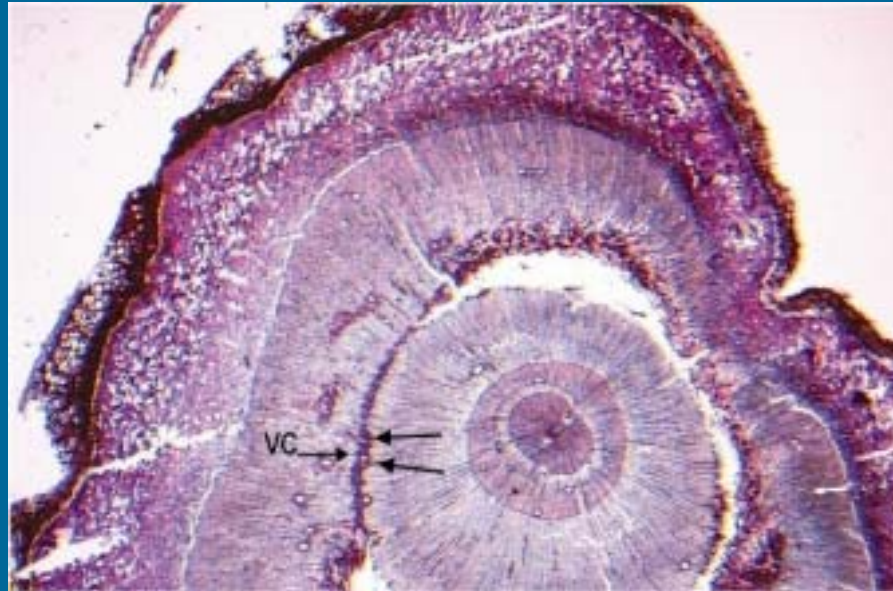
2nd ECDL Workshop on Web Archiving – 9/19/2002

Contact: Gregory.Cobena@inria.fr

Organization

- Xyleme Acquisition Architecture
 - Technical overview
 - Specific services: Site-based acquisition and focused crawling
- Online computation of Page Importance
- Xyleme Monitoring Services
 - The challenge: monitoring data on the Web
 - Technical overview
 - Sample applications: copy tracker, version management

Xylem Acquisition Architecture

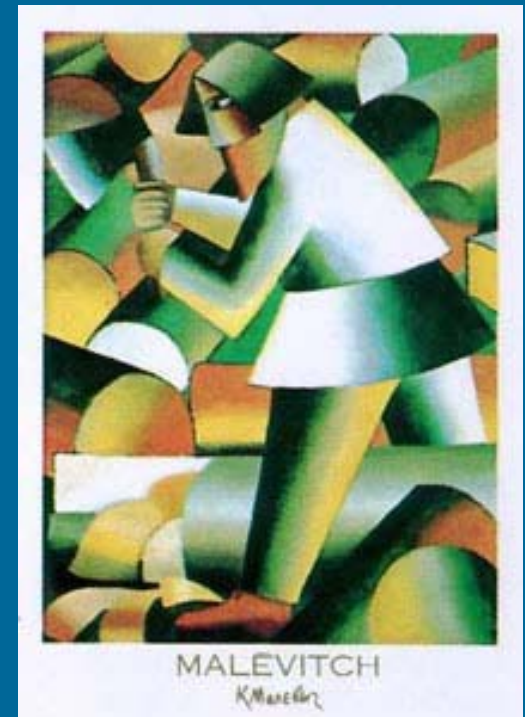


Overview

- A cluster of Linux PCs. Applications are in C++. Communications using Corba.
- The Crawler is composed of several modules:
 - The Fetcher retrieves web pages
 - The Scheduler decides which are the pages to read next
 - The Metadata Manager stores information about URLs (e.g. last date of crawl, page importance)
- We crawl about 4 million pages/PC/day on average
Each PC is in charge of 200 million URLs
- The system has been running since mid-2000. We have 8 PCs, each equipped with 2Gb of memory. We know about 1.5 billion pages.

Metadata Module

- Metadata about each URL is stored on disk
- We assign an ID to each URL
 - Stored in main memory: 1Gb for 100million URLs
 - Online page rank is also stored in memory
- We use an evaluation of the change rate of pages
 - Based on the Last Date of Change (HTTP Header)
 - Based on the number of changes that we detected



Fetcher Module

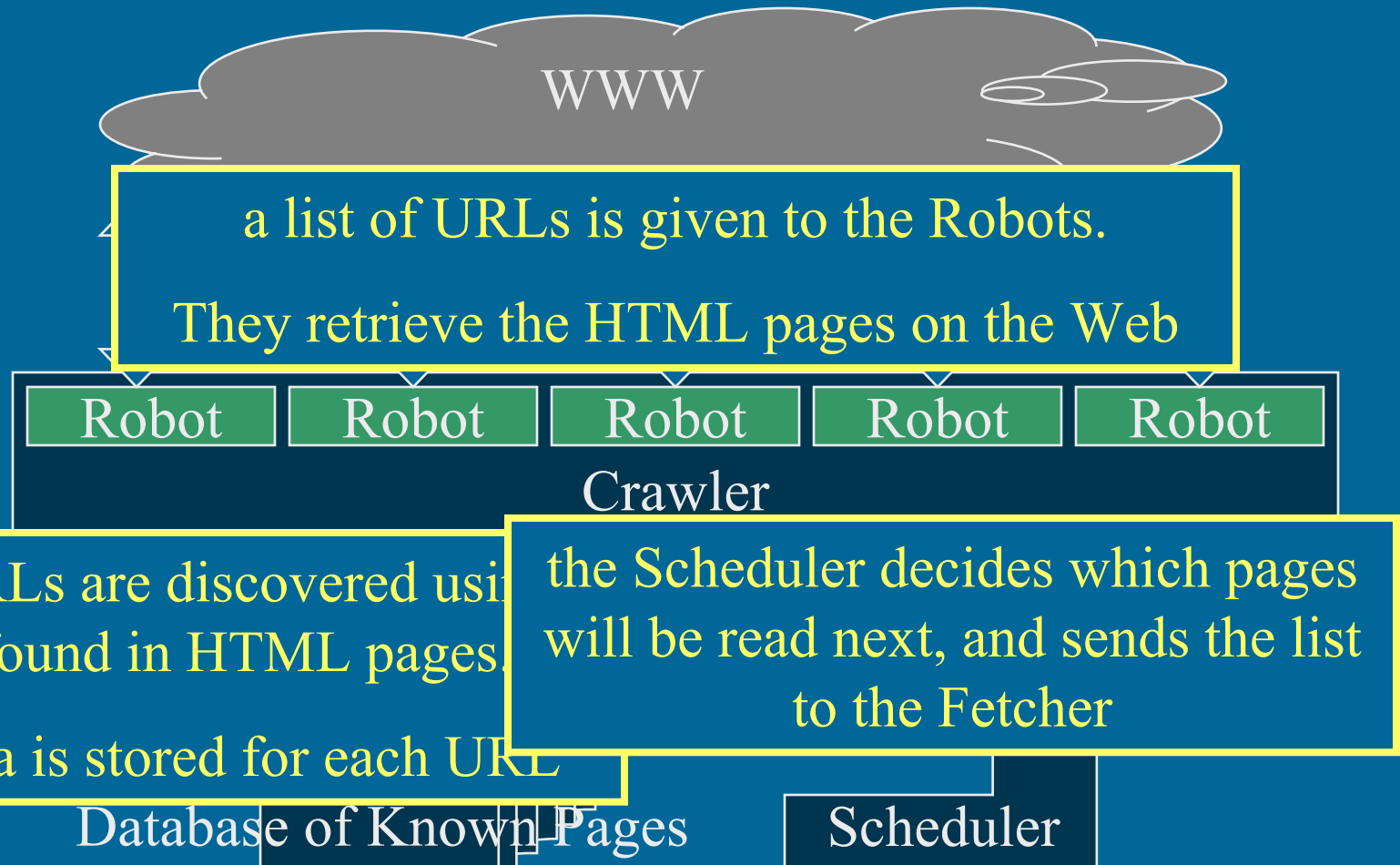
- 200 robots are running simultaneously on a single PC
 - Peak rate is 200 pages/s (e.g. 17 million/day)
 - We use persistent objects in memory to avoid rapid ‘firing’
- DNS Caching: On each PC, pages are grouped by domain to minimize the cost of DNS (Domain Name Server) resolution.
 - Currently 200.000 domain names per PC.
- To efficiently access each URL metadata, the ID table is stored in main memory:
 - 100 million URLs for 1Gb of memory
- A list of pages that are waiting to be read
 - up to 10 million

Scheduler Module (Refresh Policy)

- The goal is to balance bandwidth resources between the discovery of new pages and refreshing others
- Decision is based on the minimization of a cost function
- Refresh strategy is based on:
 - Importance of the page
 - Staleness of the page
 - Estimated change rate of the page
 - Publication or Subscription requests



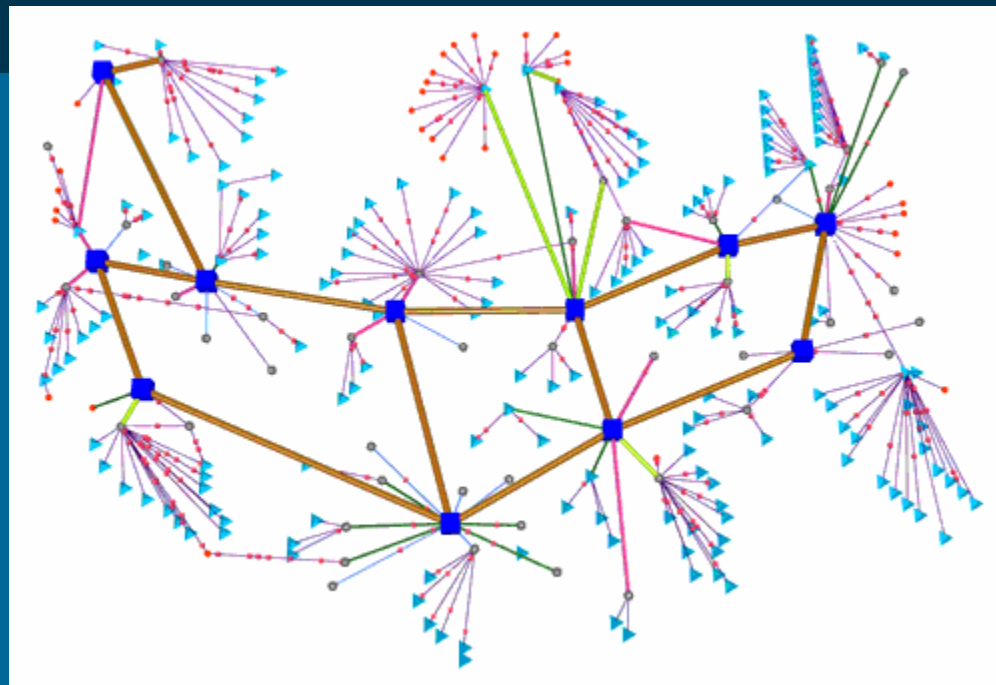
Page Life Cycle



Issues

- Site vs. Page archiving
 - The Web:
 - Physical granularity = HTML pages
 - The problem is inconsistent data and links
 - Logical granularity?
 - Snapshot view of a web site
 - What is a site?
 - There are technical issues (rapid firing, ...)
- Focused Crawling
 - We allow to publish specific documents or URLs in Xyleme (refresh frequency, access rights, ...)
 - We allow ‘on-demand’ crawling (with low latency response time)

Online computation of Page Importance



What is page importance?

- “Le Louvre” homepage is more important than an unknown person’s homepage
- Important pages are pointed by:
 - Other important pages
 - Many unimportant pages
- This leads to Google definition of PageRank
 - Based on the link structure of the web
 - used with remarkable success by Google for ranking results
- Useful but not sufficient for web archiving

Modeling the Web

- The Web as a graph: We view the WWW as a directed graph G .
 - Web pages are vertices, HTML links are edges.
- A graph as a matrix:

A graph with n vertices can be presented as a matrix $L[1..n, 1..n]$ such that:

 - L is nonnegative, i.e. $L[i, j] \geq 0$
 - $L[i, j] > 0$ if and only if there is an edge from vertex $\langle i \rangle$ to vertex $\langle j \rangle$
- There are several natural ways to encode a graph as a matrix:
 - Google defines the outdegree and set $L[i, j] = 1/\text{out}[i]$
 - Kleinberg proposes to set $K[i, j] = 1$ if there is a link from i to j and then use $L = {}^tK * K$

Importance Definition

- In short, page importance is the fixpoint X of the equation $L * X = X$
- We define the importance of a page by an inductive way and compute it using a fixpoint
- The importance is represented as a vector x in an n dimensional space.
- The equation is: $x_{k+1} = L * x_k$

Our Algorithm

- With off-line algorithms, it is necessary to store the link matrix and to compute page importance in an off-line process
- In general, off-line algorithms are started after a full snapshot-crawl of the Web
- Our algorithm:
 - Computes the page importance (characteristic vector) and does not require any assumption on the graph
 - starts even when a (large) part of the matrix is still unknown
 - helps deciding which new part of the matrix should be acquired
 - is integrated in the crawling process
 - works on-line even while the graph is being updated
- It has been presented in “Computing web page importance without storing the graph of the web (extended abstract)”, IEEE Data Engineering Bulletin, Volume 25, March 2002 ”

Static Graphs: OPIC Algorithm

- We assign to each page a small amount of 'cash'. The 'cash' is distributed among its children when the page is read.
- The page importance for a given page is found using the history of 'cash' that was stored.
- 'cash' is stored in main memory for each page
- A unique disk access per page read is necessary to update the history of the page that is read



Crawling Strategies

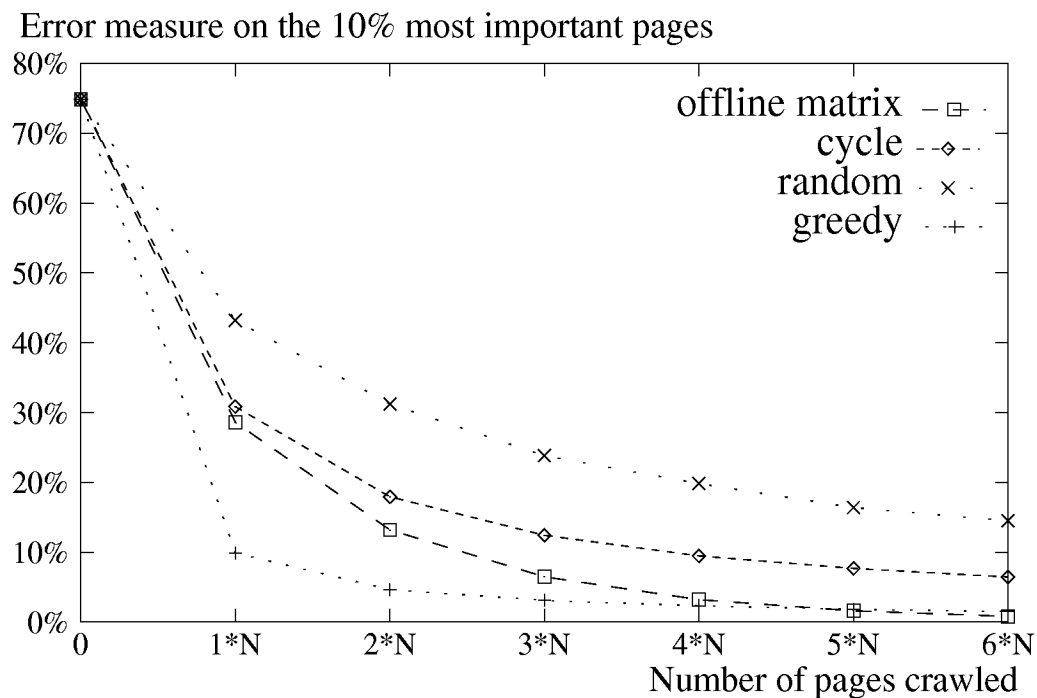
- Our algorithm works with any crawling strategy as long as each page is visited infinitely often.
It does not impose any constraints on the order of pages to visit
- Simple crawling strategies are:
 - Random: all pages have equal probability to be chosen
 - Greedy: choose the page with largest amount of ‘cash’
 - Cycle: systematic strategy that cycles around the set of pages
- The crawling strategy in Xyleme is close to Greedy since it is tailored to optimize our knowledge of the Web
- Convergence is faster with Greedy since pages have more cash on average

A changing graph

- The Web changes continuously, so does the importance of pages.
- Our adaptive algorithm works by considering only the recent part of the ‘cash’ history for each page
- The time window corresponding to the ‘recent history’ may be defined as:
 - A fixed number of measures for each page
 - A fixed period of time for each page
 - A single value that interpolates the history for a specific period of time
- When the number of nodes changes, there are some difficulties.
More precisely, the page importance of previously existing pages decreases automatically.

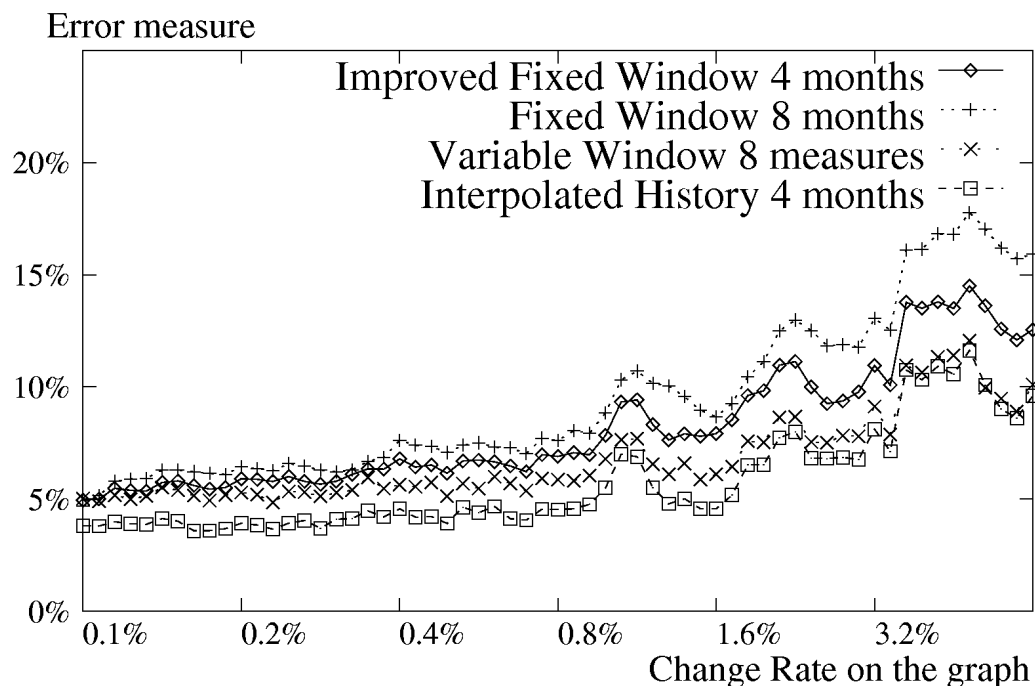
Experiments on synthetic data

- Convergence on important pages



Experiments on synthetic data

- Impact of the window policy



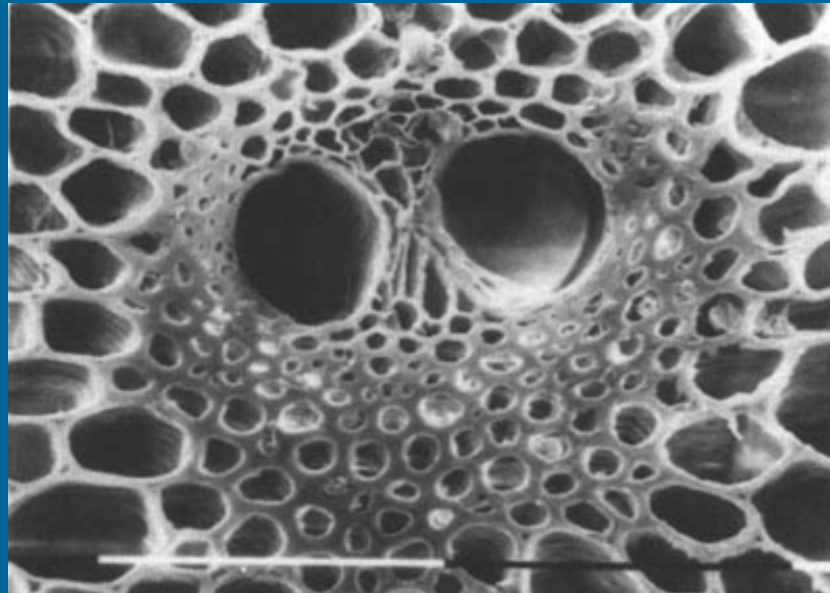
Experiments on Web data

- Experiments where conducted using the crawlers of Xyleme (e.g. 8 PCs with 1.5Gb of memory)
- Crawling strategy is close to Greedy
- History is managed using the Interpolation policy
- Experiments lasted for several months, we discovered close to one billion URLs and read 400 millions of them
- Importance of read pages seems correct (with limited human checking).
- We could also give importance estimates for pages that were never read
- The size of the window was first too small, then we set it to 3 months

New directions: site vs. pages

- Limitation of page importance
 - Google page importance works well when links have a strong semantic
 - More and more web pages are automatically generated and most links have little semantics
- More limitation
 - Refresh at the page level presents drawbacks
- So we also use link topology between sites and not only between pages

Xylem Monitoring Services



What data is there to monitor?

- Documents: HTML but also doc, pdf, ps...
- Many data exchange formats such as asn1, bibtex...
- **New official data exchange format: XML**
- Hidden web: database queries behind forms or scripts
- Multimedia data: ignored here
- Public vs. private (Intranet or Internet+passwd)
- Static vs. dynamic



What is changing?

- Universal data exchange format (**XML**) is coming
 - Universal data exchange format
 - Marriage of document and database worlds
 - Standard query language: XQuery
 - Growing very slowly on public web (less than 1%); very fast on Intranet
- Web services (**SOAP**) are coming
 - Format for exporting services
 - Format for encapsulating queries
- Semantic web is announced
 - RDF for data
 - WSDL+UDDI for services

What is not changing fast or even getting worse

- More and more data
 - More and more junk
 - More and more stale data
- The simplicity of HTML
 - An asset first, now a handicap (OK for documents not data)
 - Very primitive query mechanism (keywords)
- No real change control mechanism
 - Web user is a passive observer
 - To detect change, keep reading a page



Subscription Language

- SQL-like language based on *'atomic events'*.
- Combines the use of monitoring queries and continuous queries.
- The language can be extended by adding new types of atomic events.
- Uses the XML Query Language for continuous queries. "Querying the XML Documents of the Web", V. Aguilera, S. Cluet, F. Boiscuvier, Tech. Report

subscription myPaintings

% what are the new painting entries
in Musee d'Orsay

monitoring newPainting
select URL

where URL extends
*www.musee-orsay.fr/**

and <painter> contains "Monet"

% manage the changes

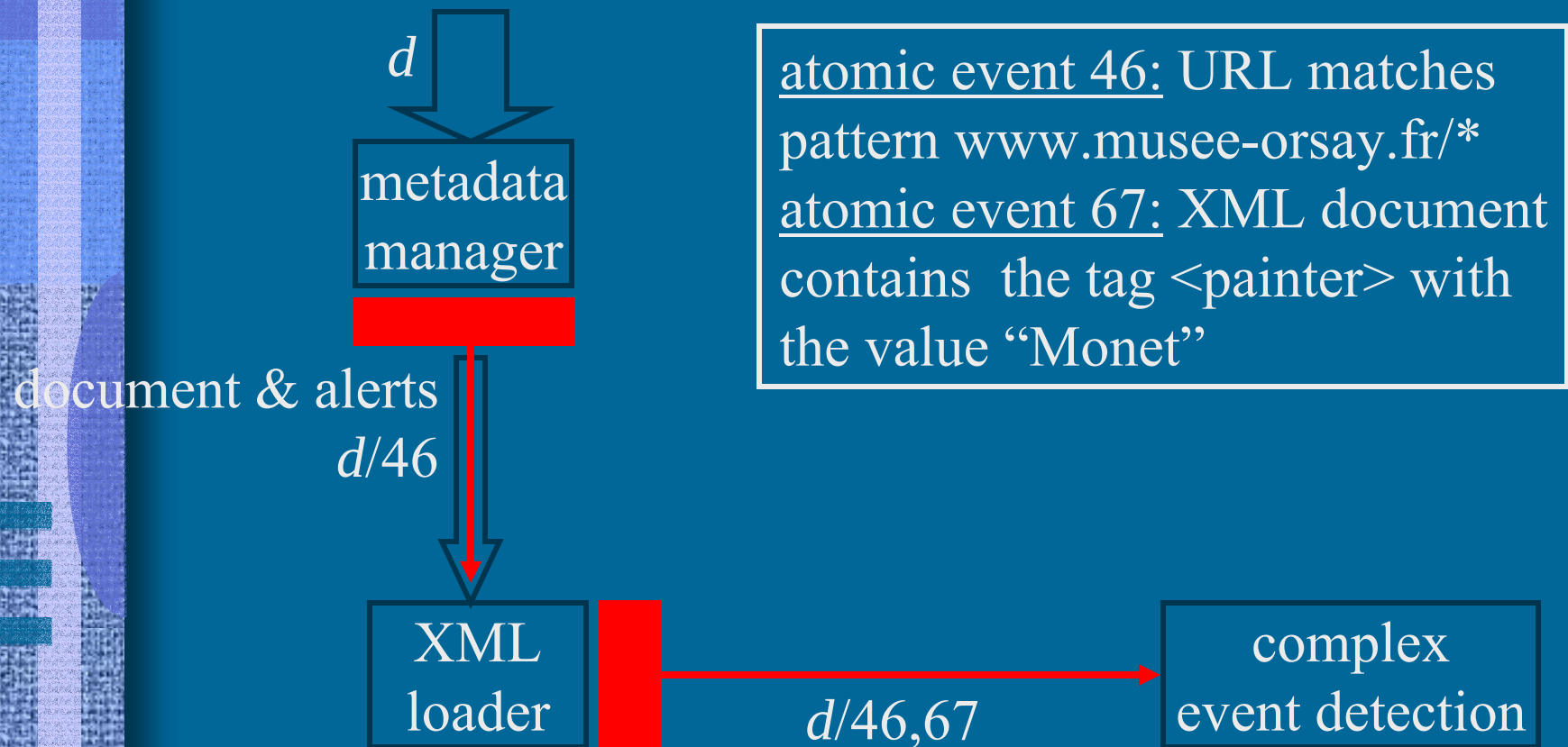
continuous delta

select ... from ... where
when monthly

notify daily % send me a daily report

Step 1: Atomic Event Detection

5 millions of pages/day

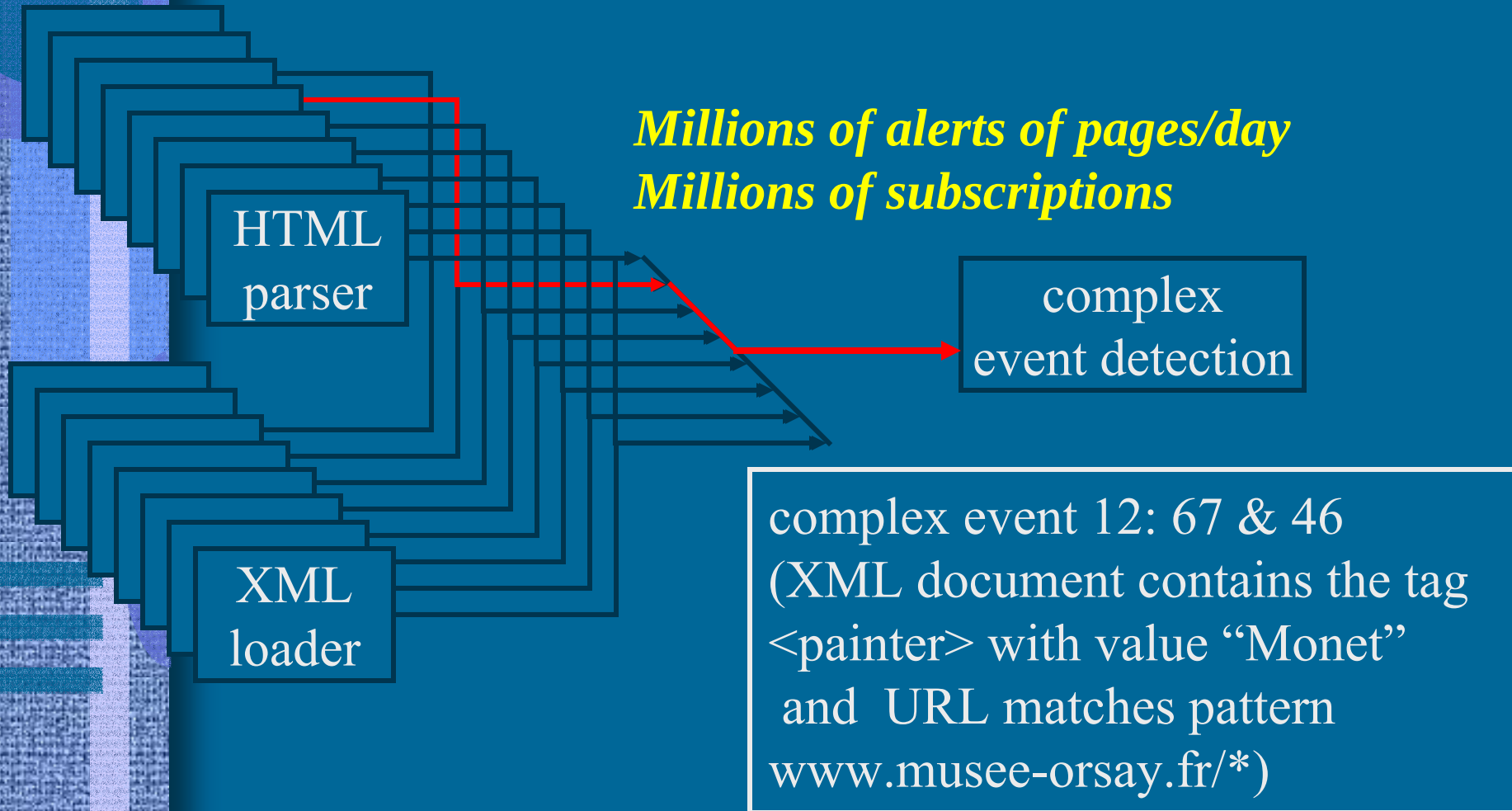


Alerters

- Each Alerter can be viewed as a plug-in that acts on a document flow.
- All sorts of Atomic events can be detected: URL pattern detection, Keywords, XPath expressions, Page rank...
- Can be distributed.
- Some advanced alerts are:
 - Long string look-ups
 - Finding XML Patterns (e.g. XPath)
 - Comparing digital signature of text documents (copy tracker)



Step 2: Complex Event Detection

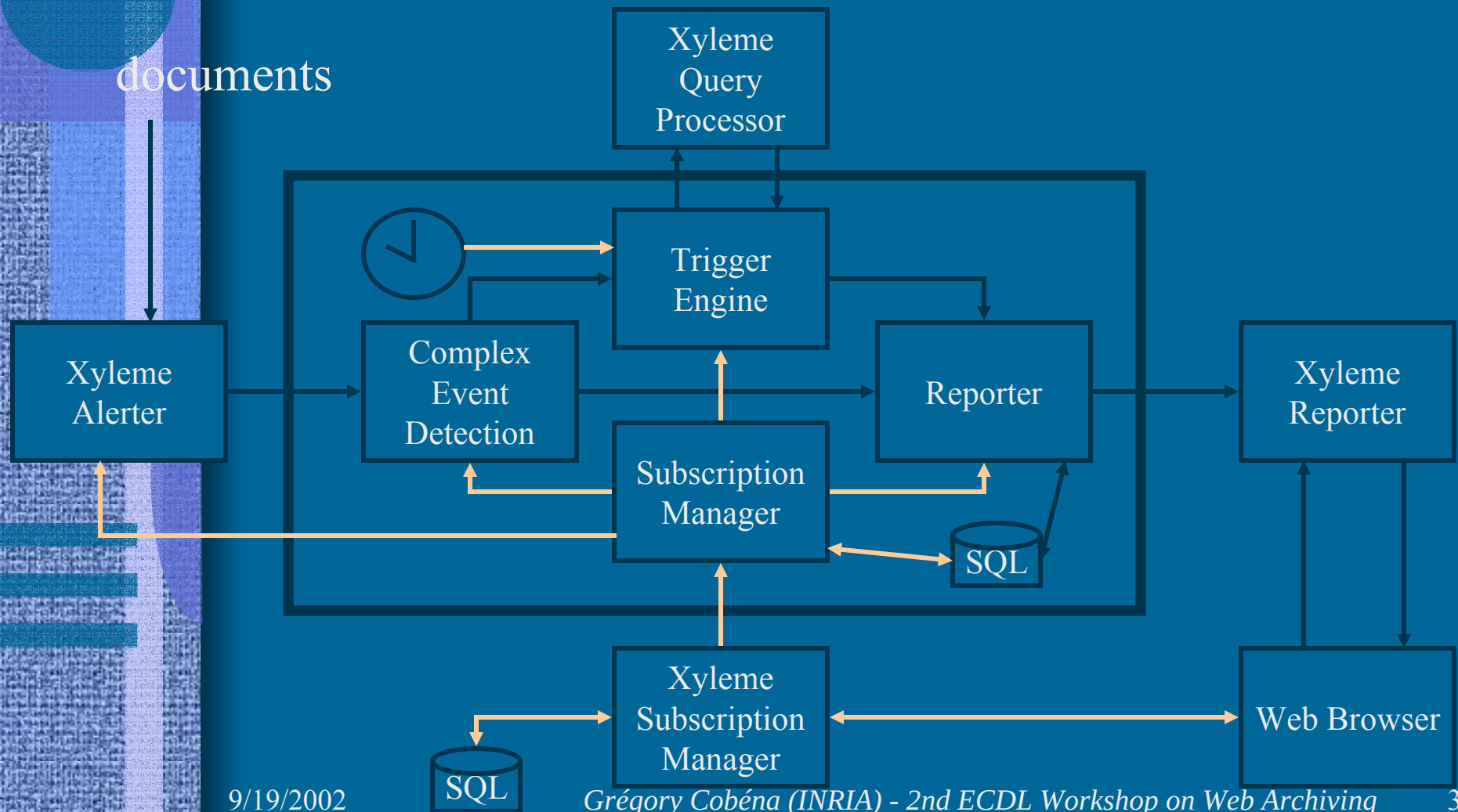


Complex Events Algorithm

- The formal problem is NP-hard
 - We proposed several possible algorithms
 - Experimental (simulation) values proved the effectiveness of our solutions
 - The Hash-Tree based algorithm is well suited for our application:
 - 10 million Complex Events
 - 1 million Atomic Events
 - 100 Atomic events detected per document
- 0.8 ms to process a document, ~2 million documents per day (on each PC).



Architecture



Monitoring Applications

- Web surveillance
 - Anti-criminal intelligence, business intelligence
 - Classification, extracting data of interest, extracting knowledge
- Copy tracking
 - a press agency wants to check that people are not publishing illegally copies of their wires
- Web portal management
 - Dangling pointers, unreachable pages, incorrect pages, ...
 - Archiving, version management, temporal queries
 - monitoring changes, subscription and notification, ...

Web archiving

- We discussed an experience in archiving the French web
- Monitoring is used to increase librarians knowledge of the Web
- Monitoring is also used for specific applications such as detecting deep-Web sources (e.g. http forms)

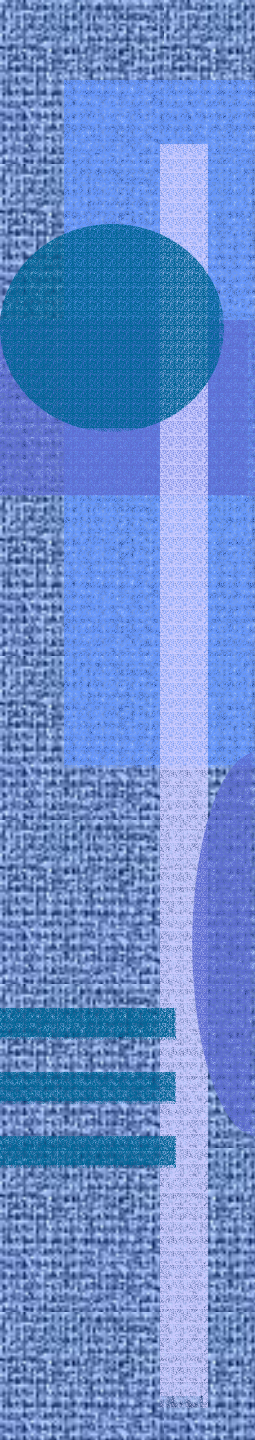


Conclusion

- Very challenging problem
 - Complexity due to the volume of data
 - Complexity due to heterogeneity
 - Complexity due to lack of cooperation from data sources
- Other issues:
 - Hidden or deep Web, ...

New directions

- Site based architecture
 - Site-based crawling
 - Site-based archives
 - Site-based importance
- Active web sites
 - Friendly sites willing to cooperate
 - Web services provide the infrastructure
 - Support for triggers



Thanks