

# Sindice.com: A lookup index for Semantic Web resources

**Eyal Oren** and Giovanni Tummarello

Digital Enterprise Research Institute  
National University of Ireland, Galway

6th June 2007

## Introduction

Sindice overview

Sindice details

Scalability

Summary

# Motivation



There is a lot of Semantic Web data

# Motivation



## Where to find statements about resources?

- ▶ Linked data is only a partial solution: would you only read reviews linked from Nokia?

# Motivation



## Where to find statements about resources?

- ▶ Linked data is only a partial solution: would you only read reviews linked from Nokia?
- ▶ We need a simple lookup index telling me where a resource has been mentioned: Sindice.com

# Sindice.com



(Sindice is not an end-user service but a programmatic API)

Introduction

**Sindice overview**

Sindice details

Scalability

Summary

## Design principles

- ▶ Acts only as locator: return pointers to remote sources
- ▶ Not a query engine: no lookups on triple patterns or joins
- ▶ Subjective/difficult things on client-side:
  - ▶ Integrate into Tabulator, Disgo, DBin, PiggyBank
  - ▶ They handle trust, entity consolidation, reasoning

## Design principles

- ▶ Acts only as locator: return pointers to remote sources
- ▶ Not a query engine: no lookups on triple patterns or joins
- ▶ Subjective/difficult things on client-side:
  - ▶ Integrate into Tabulator, Disgo, DBin, PiggyBank
  - ▶ They handle trust, entity consolidation, reasoning
- ▶ **Repeat:** we are not Swoogle or SWSE
  - ▶ We do not store triples
  - ▶ We do not answer queries
  - ▶ We do not perform entity consolidation
  - ▶ Scalability, simplicity, focus
  - ▶ Decentralised data: do as much as possible client-side

## Sindice design

Functional requirements:

1. Parse file and SPARQL endpoint when “pinged” or crawling
2. Lookup resources (URI) and return mentioning sources (URLs)
3. Lookup text (keyword) and return mentioning sources (URLs)

Non-functional requirements:

1. Scale to at least 300 million resources
2. Minimise index size
3. Minimise lookup time
4. Continuous live index updates

Introduction

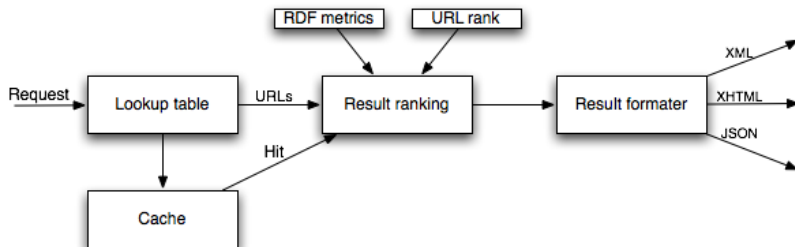
Sindice overview

**Sindice details**

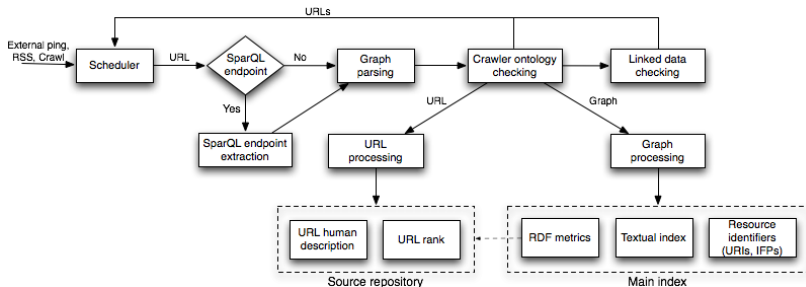
Scalability

Summary

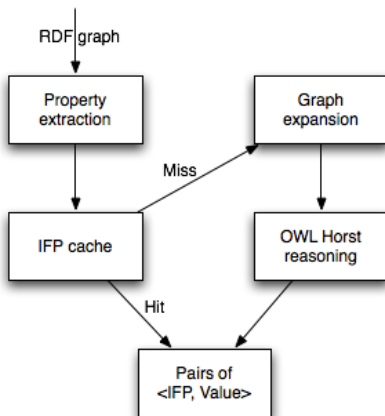
## Querying pipeline



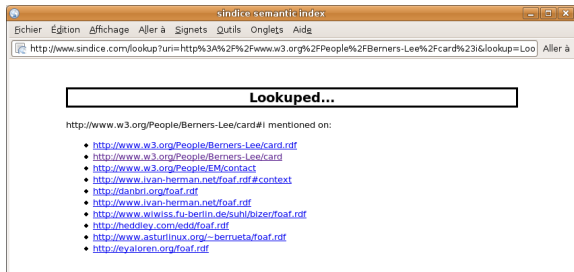
## Indexing pipeline



## Graph processing



## Ranking phase



- ▶ Hostname: prefer w3.org
- ▶ External rank: prefer w3.org
- ▶ Relevant statements: prefer card.rdf
- ▶ Source size: prefer ivan-herman.net

## Index design

- ▶ Reverse lookup index: *resource*  $\rightarrow$  *source*<sup>+</sup>
  - ▶ Database (mysql, sqlite): 38–100GB main memory
  - ▶ Persistent hashtable (bdb, qdbm): poor scalability
  - ▶ File system (ext3, reiserfs):
    - ▶ use hash(URI) as filename, file content is list of sources
    - ▶ lookup(URI): read file, convert to requested format
    - ▶ but max. #files: partition hash into subdirs
    - ▶ but minimal block-size: use reiser4
- ▶ Source metadata: *source*  $\rightarrow$  *properties*

Introduction

Sindice overview

Sindice details

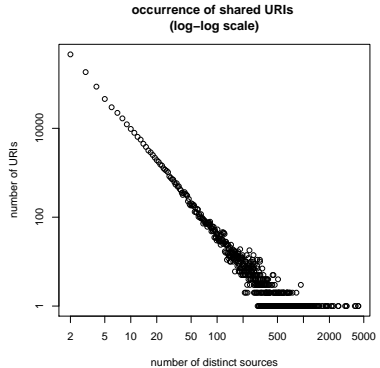
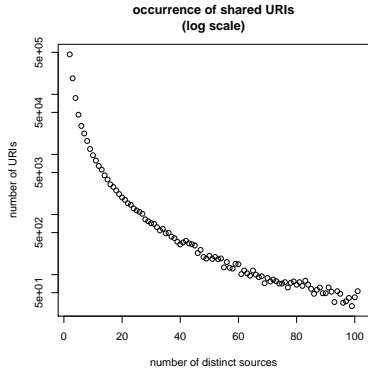
**Scalability**

Summary

## Feasibility

- ▶ Crawled `pingthesemanticweb.com` for four weeks
  - ▶ Collected 3.2 million unique URIs (SWSE: 11 million)
  - ▶ Required 2.5GB index space
- ▶ Now we want to extrapolate to 300 million resources:
  - ▶ Index size depends on how often resource is mentioned
  - ▶ URI mentioning follows power-law (scale-free)
  - ▶ Extrapolate index size linearly with number of resources
  - ▶ 300 million resources: 785GB index size

## Feasibility: URI reuse



Introduction

Sindice overview

Sindice details

Scalability

Summary

## Current status

- ▶ Live at <http://sindice.com>
  - ▶ Open-source (LGPL)
  - ▶ Rails web application, 200 lines of Ruby code
  - ▶ Nginx web server, Mongrel application server
  - ▶ Reiser4 on-disk index and sqlite3 for source metadata
- ▶ Fetch and extract:
  - ▶ Update sources from [pingthesemanticweb.com](http://pingthesemanticweb.com)
  - ▶ Fetch RDF using Redland rapper
  - ▶ Extract URIs using regular expressions
- ▶ Query and use:
  - ▶ HTTP API with XML, JSON, HTML, txt results