Développement des Algorithmes d'Application Réticulaire

https://www-apr.lip6.fr/~buixuan/daar2021

Binh-Minh Bui-Xuan

PARIS, December 2021

Lecture 10 : Indexing and architecture

Recall lecture + tme 1-2 :

- Searching from desktop
- Desktop → unique machine
- Unique memory → RegEx and document as arguments

Lecture 10 : Indexing and architecture

Recall lecture + tme 1-2 :

- Searching from desktop
- Desktop → unique machine
- Unique memory → RegEx and document as arguments

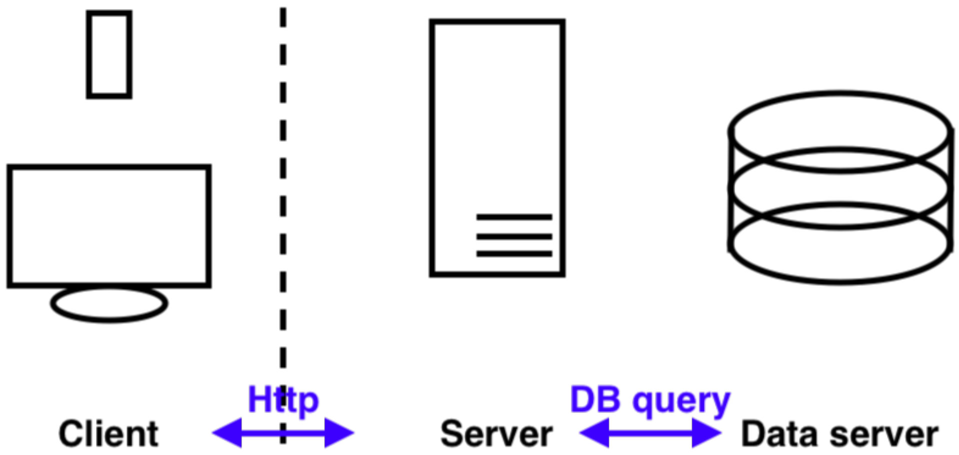
Today work :

- Searching online
- Online : client-server model
- Online : real time response → indexing
- Monolithic vs. microservice architectures

Part I. Client-server by API

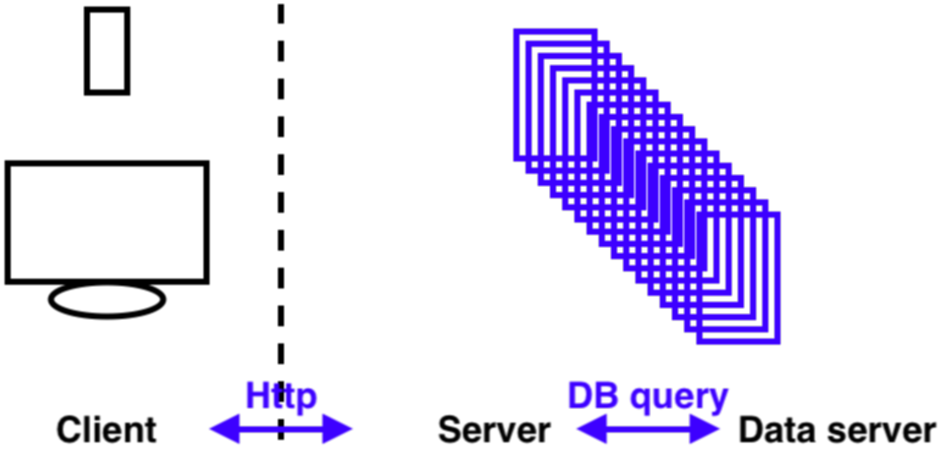
Example with a librarian webapp

Client-server model : three layers



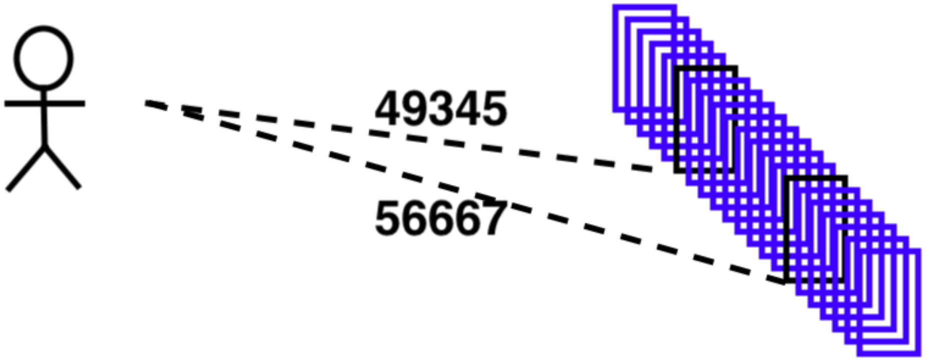
- Numerous client machines → Http traffic
- Some servers (controllers if MVC ; brookers ; load balancers ; gateways ; ...)
- Fewer data servers (on premises) or numerous data servers (on cloud)

Client-server model : three layers



- API view : response in standardized format
- Example : `http://www.gutenberg.org/ebooks/` → list of all books
- `http://www.gutenberg.org/ebooks/id` → details of book with `pk` equals `id`

API query ?

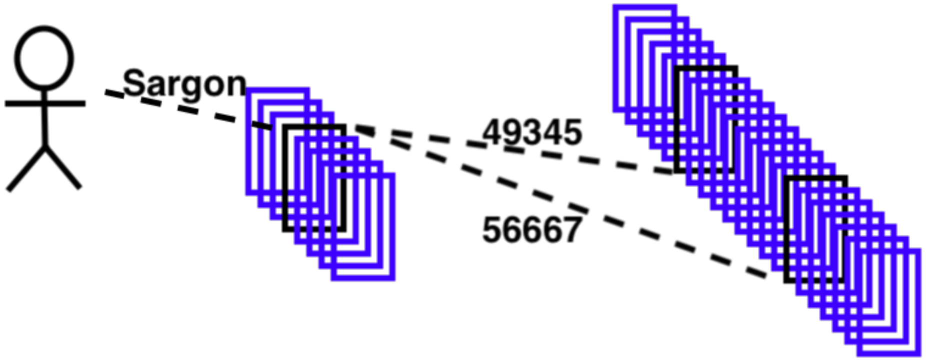


– User asks `49345` ; Gutenberg responses ***History of Babylonia vol. 1***

– User asks `56667` ; Gutenberg responses ***History of Babylonia vol. 2***

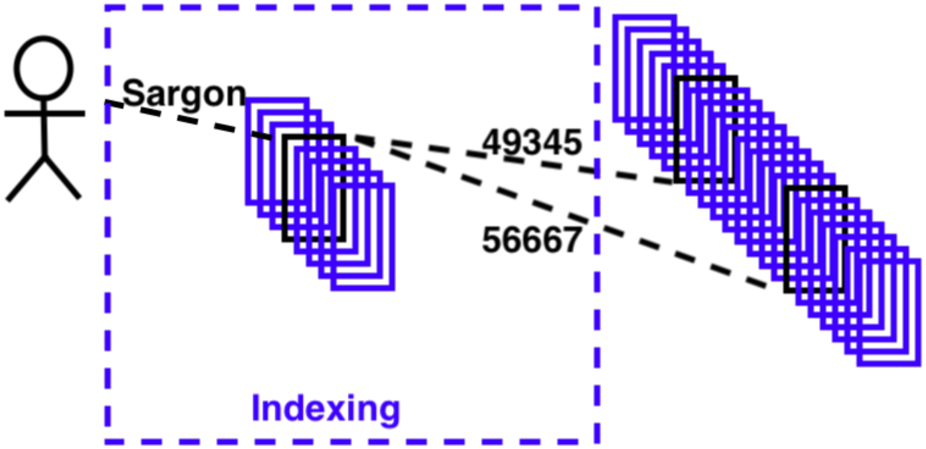
– Issue : meaning of `49345` and `56667` to a human ? !

API query ? Redirection ?



- Want : user asks `Sargon` for a list of relevant books, e.g. `49345` and `56667`
- Choice : middle man server vs. client-side computation ?
- Issue : computing time ? precomputing ? real-time response ?

API query ? Redirection ? Indexing ?



- Indexing : precompute part of the answers, e.g. keywords, names, ...
- Wiring : each precomputed answer links to the original document
- Issue : space explosion ? automatic update ?

Indexing a textual document

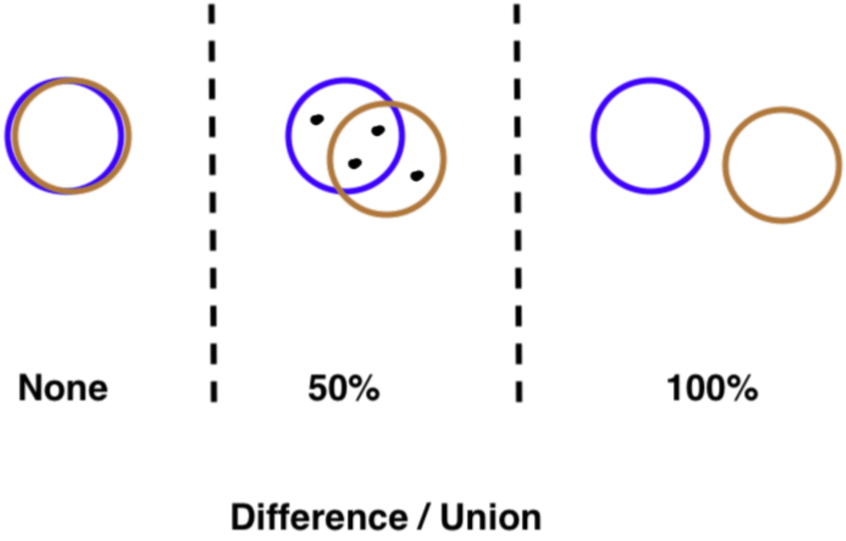
Naive table :

- Document D is considered as a list $l(D)$ of words split by `[^A-Za-z]`
- N.B. : hence, the base alphabet is `[A-Za-z]`, i.e. the Latin alphabet
- The index table is $l(D)$
- N.B. : $l(D)$ can be trimmed off 1-character and 2-characters words

Weighted naive table :

- Each index is now (w, k) representing word w has k occurrences in D
- The weighted index table is the set of all such pairs (w, k)

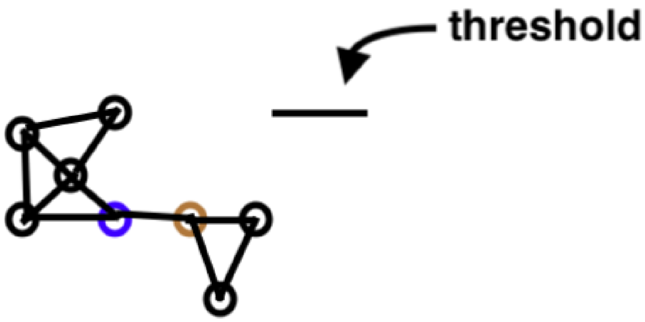
Jaccard distance between index tables



Distance between two documents based on their index tables :

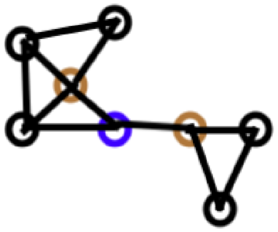
$$d(D_1, D_2) = \frac{\sum_{(w,k_1)\in D_1 \wedge (w,k_2)\in D_2} \max(k_1, k_2) - \min(k_1, k_2)}{\sum_{(w,k_1)\in D_1 \wedge (w,k_2)\in D_2} \max(k_1, k_2)}$$

Geometric graph



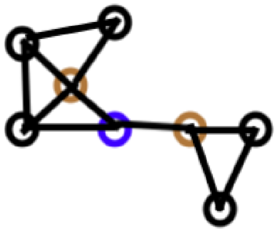
- Vertex set : set of all documents
- Parameter : threshold distance, e.g. $\theta = 50\%$
- Edge set : (D_1, D_2) is an edge iff $d(D_1, D_2) < \theta$

Centrality ranking



- Question : what is the most recommendable book ?
- Having many similar books ? → vertex of highest degree ?
- Close to all books ? → center vertex, closeness centrality
- Middle man of many books ? → betweenness centrality

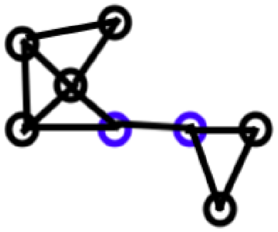
Closeness centrality



Ranking of vertices by closeness centrality :

$$crank(v) = \frac{n-1}{\sum_{u \neq v} d(u,v)}$$

Betweenness centrality



Ranking of vertices by betweenness centrality :

$$brank(v) = \sum_{s \neq v \neq t} \frac{\#paths(s,t,v)}{\#paths(s,t)}$$

where $\#paths(s,t)$ is the number of shortest paths from s to t ; and $\#paths(s,t,v)$ is the number of shortest paths from s to t passing through v.

Part II. Breaking the (futur) monolith

Example with an image service

Philosophical question

What is image ?

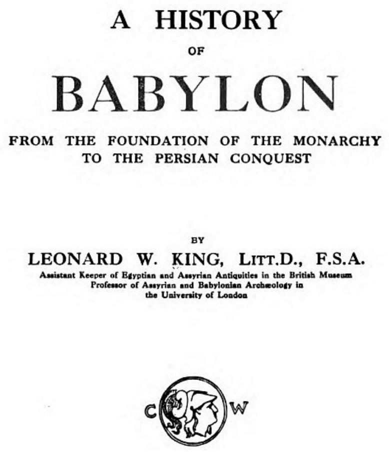
– Me on Instagram ?

– A moment of real life ? (captured by some means ?)

– ***Artifact that depicts visual perception.*** (Wikipedia)

– An URL, e.g. `http://gutenberg.org/files/56667/56667-h/images/cover.jpg`

Our assumption



A HISTORY
OF
BABYLON
FROM THE FOUNDATION OF THE MONARCHY
TO THE PERSIAN CONQUEST

BY
LEONARD W. KING, LITT.D., F.S.A.
Assistant Keeper of Egyptian and Assyrian Antiquities in the British Museum
Professor of Assyrian and Babylonian Archæology in
the University of London

- An image location is an URL.
- An image resource is a file stored in the file system (PNG, JPG, etc).
- An image can be associated to an id, e.g. #56667.

What we expected after part I

Part I webapp is to serve :

- `.../book/<id>/` → details of the book with `pk` egals `id`.
- `.../frenchbook/<id>/` → details of the French book with `pk` egals `id`.
- `.../englishbook/<id>/` → details of the English book with `pk` egals `id`.

What we expect in part II

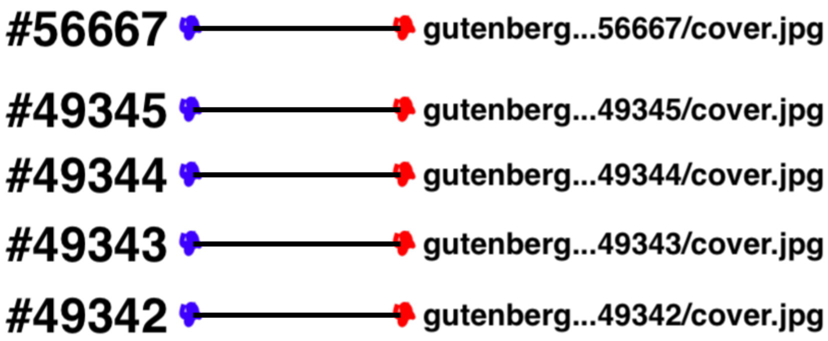
Part I webapp is to serve :

- `.../book/<id>/` → details of the book with `pk` egals `id`.
- `.../frenchbook/<id>/` → details of the French book with `pk` egals `id`.
- `.../englishbook/<id>/` → details of the English book with `pk` egals `id`.

Part II webapp is to serve :

- `.../book/<id>/coverImage` → URL of the cover image of the book with `pk=id`.

Naive image indexing



#56667 gutenberg...56667/cover.jpg

#49345 gutenberg...49345/cover.jpg

#49344 gutenberg...49344/cover.jpg

#49343 gutenberg...49343/cover.jpg

#49342 gutenberg...49342/cover.jpg

– *Search engine optimisation indexing collects, parses, and stores data to facilitate fast and accurate information retrieval.* (Wikipedia)

– Okay, so basically the best index is the `pk`! Fastest and most accurate.

– Naive indexing : perfect matching between image `id` and document `pk`.

– Eurêka : one-to-one correspondance between a book and its cover image!

Yet another philosophical question

PS : this is also the last slide !

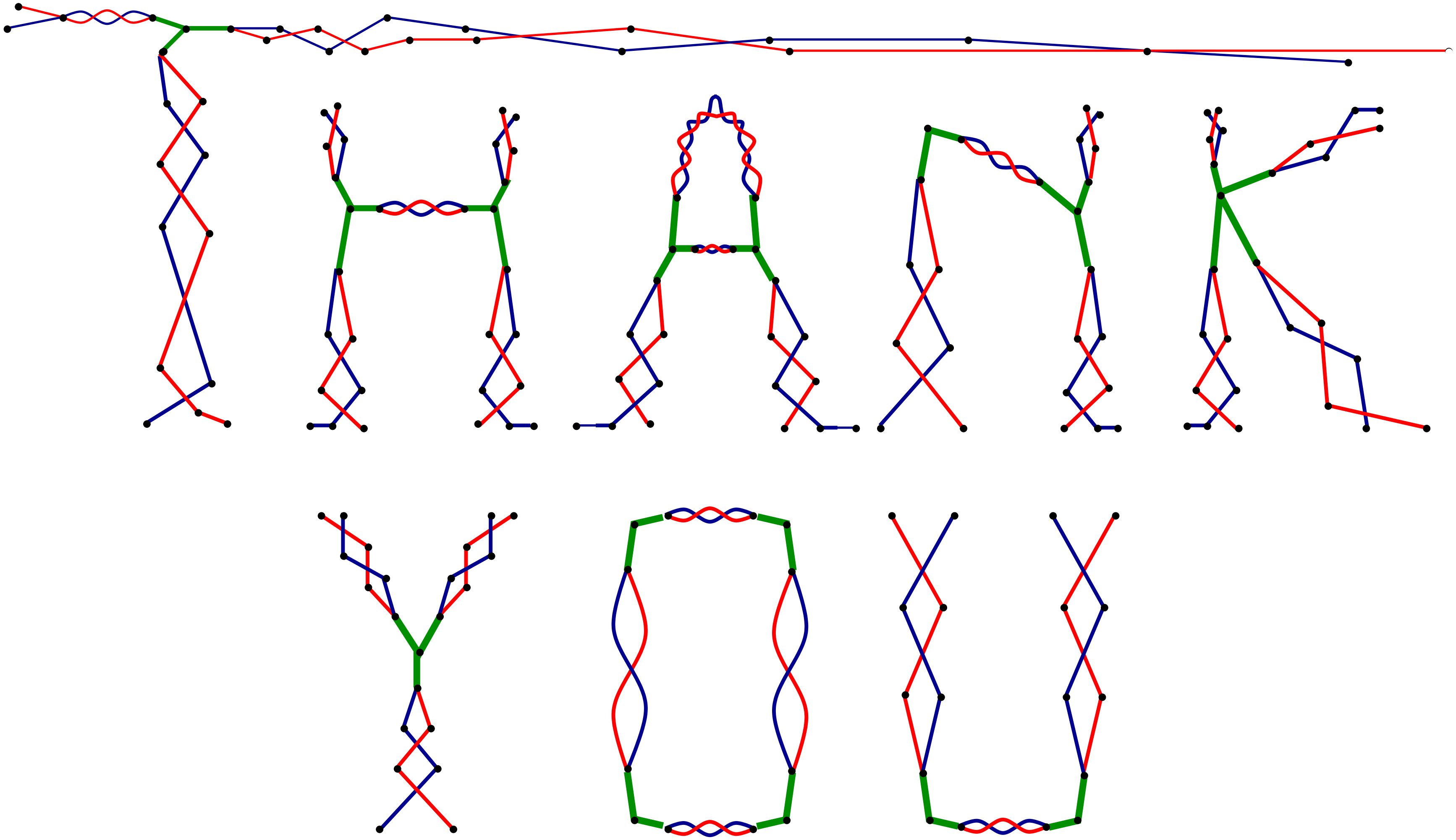
Extend part I's code ? Start new webapp ?
(1) (2)

- Pros (1) : quick, fewer code, no external resourcing, no reverse lookup, ...
- Pros (2) : continuous refactoring, versatility, new app serves many apps, ...
- Cons (1) : monolith kills : burnout of the Devs, publish or perish, ...
- Cons (2) : microservices kill : fault tolerance, must DevSecOps, ...

Conclusion

- Client-server model with API.
- An image location is an URL.
- An image file is what is actually stored in the file system.
- There are pros and cons of monolith and microservices.

⇒ enjoy TME10 practical works !


THANK
YOU