Science des données et **fouille** du WEB
Master Informatique
AIDN
2020-2021

Pierre-Francois.Marteau@univ-ubs.fr

Objectifs du module

- Anatomie d'un moteur de recherche
  - Structure du web
  - Concepts théoriques et points durs
  - Algorithmiques et modèles de traitement de l'information
  - Architecture logicielle et matérielle
- Mise en œuvre pratique
  - Conception et réalisation d'un moteur de recherche sur flux RSS

Organisation du module

- Organisation
  - 10 semaines
  - ~16h de Cours magistral
  - ~24h de TD/TP pour la partie mise en œuvre
- Contrôle des connaissances
  - ½ Contrôle Continu (mise en œuvre)
  - ½ Contrôle Terminal

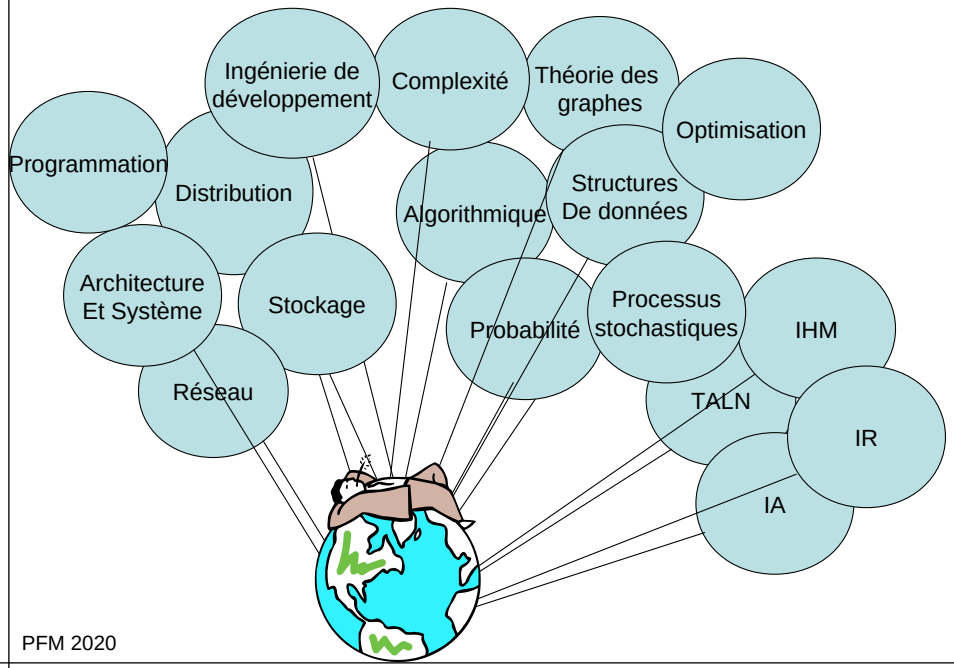
Mise en œuvre (TD/TP)

- Groupes de 2 étudiants
- Remise des rapports d'avancement à échéance fixée
- Remise du code développé et des manuels d'utilisation et d'installation
- Remise d'un rapport final de synthèse
- *Plagia non acceptable (citation des sources, licences et droit d'exploitation, … indispensable)*

Sommaire du CM

- Introduction : Principes et Architecture d'un Moteur de Recherche sur le web
- Les robots de collecte d'information (spider, crawler)
- Le stockage des pages web
- Quelques modèles d'indexation
- Analyse de liens et évaluation de l'importance des pages
- Quelques modèles de recherche d'information
- Classification des documents
- Evaluation des performances
- *La recherche de documents non textuels*
  - *Les séries temporelles*
  - *Les images*
- *Le WEB Invisible*

Champs disciplinaires



Pour en savoir plus

- The second edition of *Managing Gigabytes: Compressing and Indexing Documents and Images* by Ian H. Witten, Alistair Moffat, and Timothy C. Bell, is now available (May 1999), published by Morgan Kaufmann Publishing, San Francisco, ISBN 1-55860-570-3.
- The Anatomy of a Large Scale Hypertextual Web Search Engine (Google architecture) http://www-db.stanford.edu/pub/papers/google.pdf
- Searching The Web, Arasu, Arvind; Cho, Junghoo; Garcia-Molina, Hector; Paepcke, Andreas; Raghavan, Sriram, Stanford http://dbpubs.stanford.edu:8090/pub/2000-37
- Mercator: A Scalable, Extensible Web Crawler by Allan Heydon and Marc Najork http://research.compaq.com/SRC/mercator/papers/www/paper.html
- 1. C. J. van RIJSBERGEN, INFORMATION RETRIEVAL, http://www.dcs.gla.ac.uk/Keith/pdf/ (1977)
- Information Retrieval data Structures & Algorithms, Bill Frakes, Ricardo Baeza-Yates ed.
- Le WEB …

Sommaire du CM

⇨ Introduction : Principes et Architecture d'un Moteur de Recherche sur le web

- Les robots de collecte d'information (spider, crawler)
- Le stockage des pages web
- Quelques modèles d'indexation
- Analyse de liens et évaluation de l'importance des pages
- Quelques modèles de recherche d'information
- Classification des documents
- Evaluation des performances
- La recherche de documents non textuels
  - Les séries temporelles
  - Les images
- Le WEB Invisible

Some facts on the web

The first-ever website (info.cern.ch) was published on August 6, 1991 by British physicist Tim Berners-Lee while at CERN, in Switzerland. [2] On April 30, 1993 CERN made World Wide Web ("W3" for short) technology available on a royalty-free basis to the public domain, allowing the Web to flourish.[3]

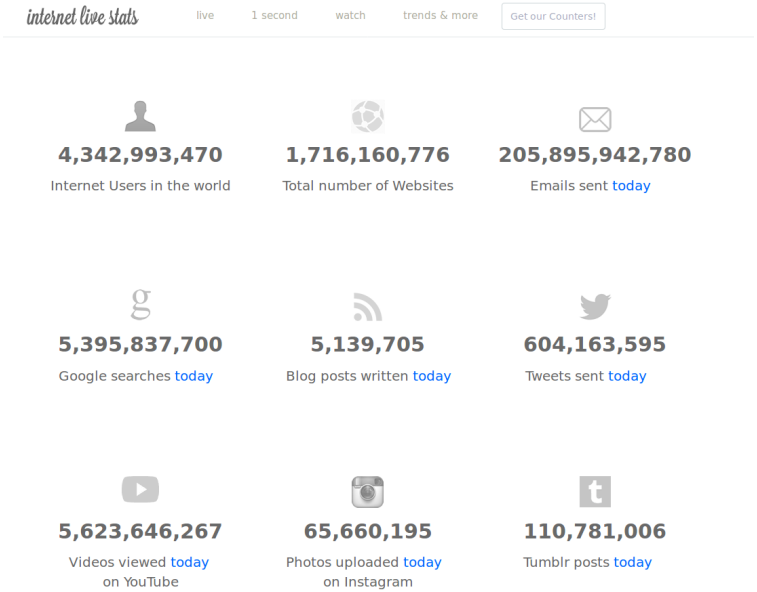
The World Wide Web was invented in March of 1989 by Tim Berners-Lee. He also introduced the first web server, the first browser and editor (the “WorldWideWeb.app”), the Hypertext Transfer Protocol (HTTP) and, in October 1990, the first version of the "HyperText Markup Language" (HTML).

In 2016, the number of websites has almost doubled: from 900 million to 1.7 billion. However, the more reliable active website count was stable at around 238 million throughout the year 2019 (www.netcraft.com).

Over 50% of websites today are hosted on either Apache or nginx, both open source web servers. However, if the trend continues, **Microsoft could soon become the leading web server developer for the first time in history**.

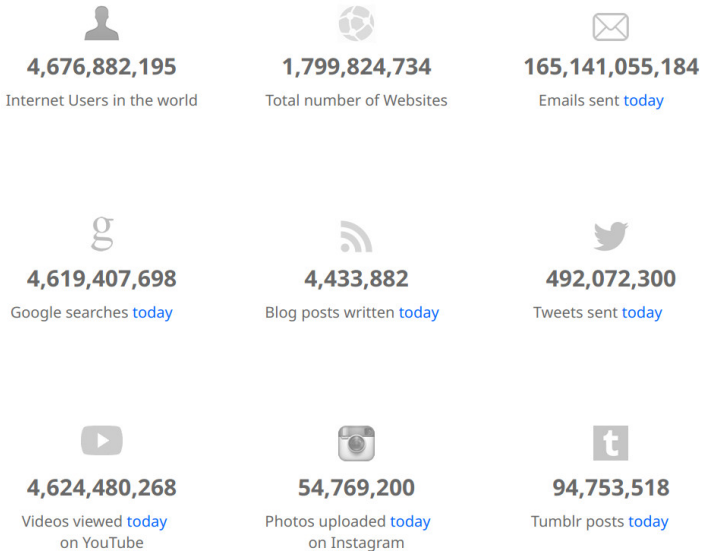
Internet Live Stats (sept. 2019):

http://www.internetlivestats.com/



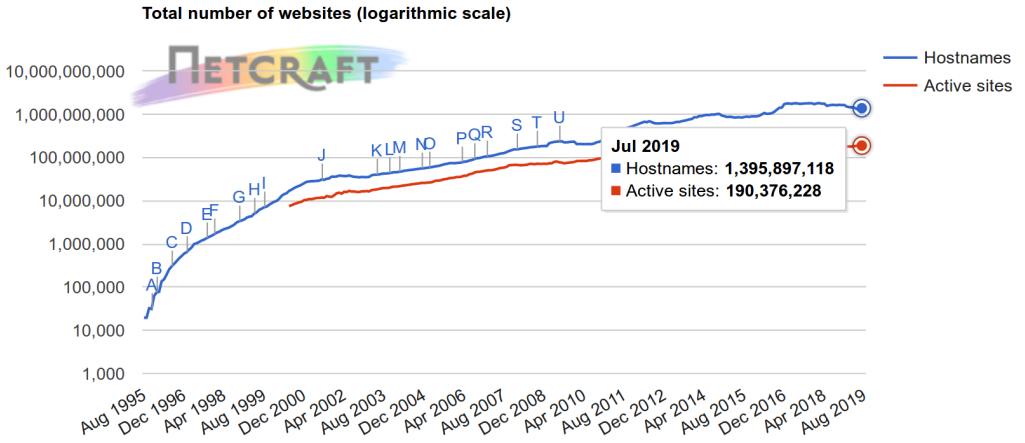
Internet Live Stats (sept. 2020):

http://www.internetlivestats.com/



Netcraft

https://news.netcraft.com/archives/category/web-server-survey/



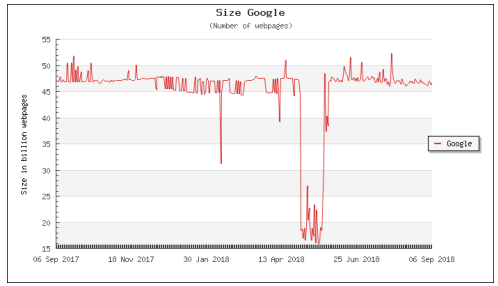
Développement web en 2020-25

- **Top 5 IoT Statistics**:
  - 26.66 billion IoT devices in 2019 were active
  - By 2025 there will be 75 Billion IoT devices in the World
  - 127 new devices are connected every second to the internet
  - By 2020, 40% of IoT devices will be used in the healthcare industry
- **L'internet contiendrait entre 1.7 et 4.5 milliards de sites web indexés** par les moteurs de recherche. Néanmoins, ce nombre colossal ne représente qu'une toute petite de ce qui existe sur le web.
- On estime que le **WEB indexé ne représente que 1 %** des données accessibles sur le web (**le « deep web » représente donc 99 % des données)** !

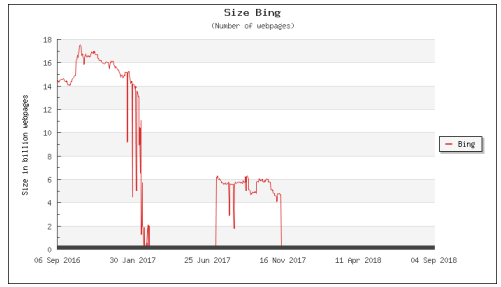
Taille du web indexé (2018)

http://www.worldwidewebsize.com/

Google ~ 47 milliards de pages



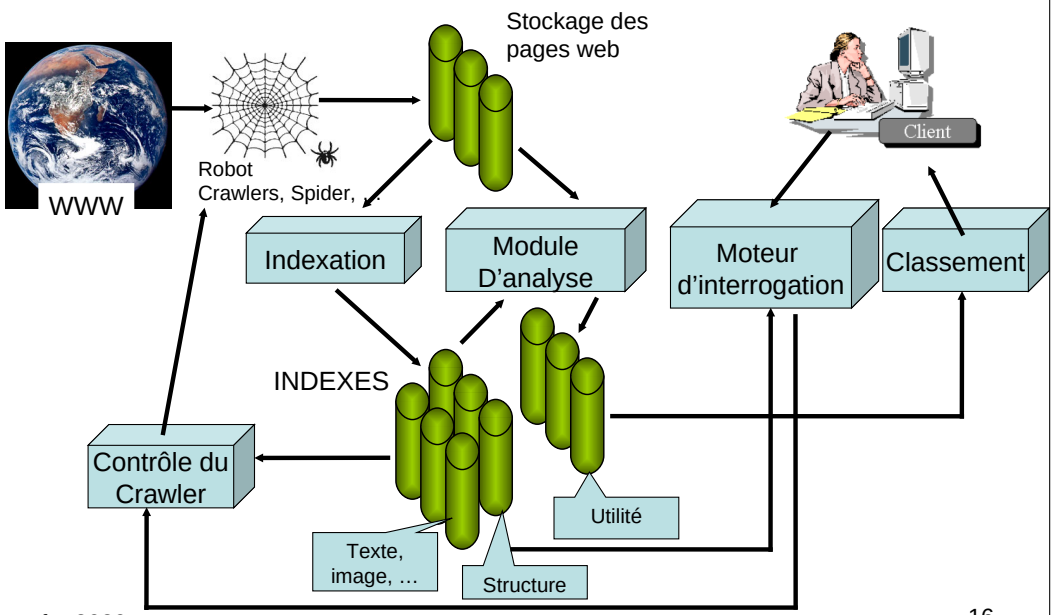
Bing ~ 170 millions de pages



Quelques points durs (parmi d'autres)

- Passage à l'échelle (scalability)
- Fiabilité
- Interopérabilité
- Obsolescence, rafraîchissement de l'information
- Gestion des flux (stream, feed, etc.).

Architecture simplifiée d'un moteur de recherche sur le WEB



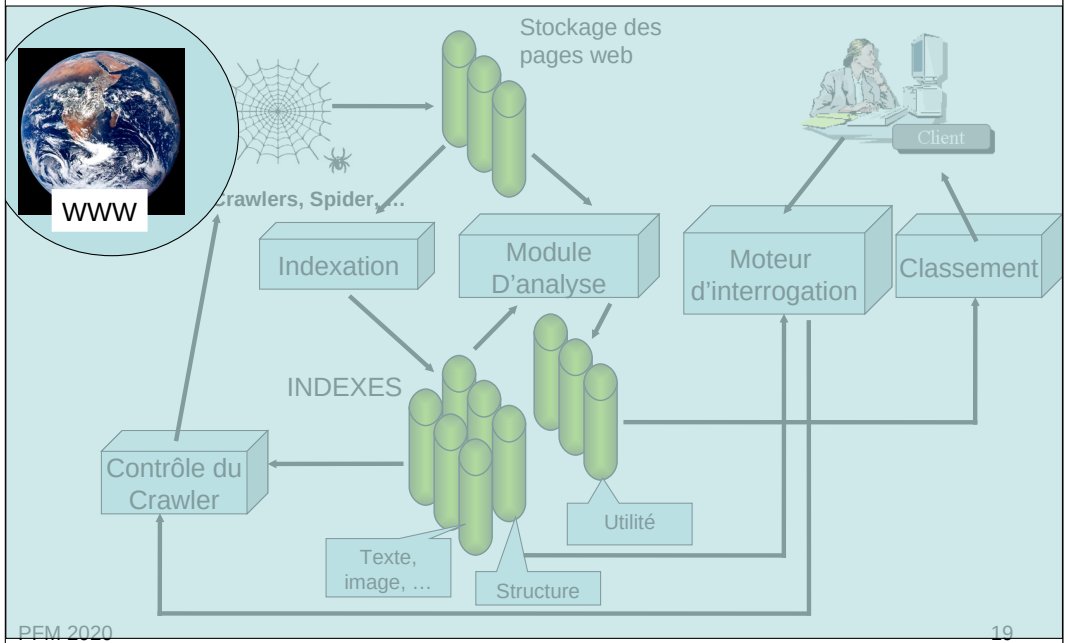
Les principaux composants

- Les robots de collecte (crawler, spider)
  - parcourent la toile (récursivement)
  - extraient l'information contenue dans les pages
- Le stockage des pages web
- Les systèmes d'indexation des contenus et de la structure des pages web
- Le module d'analyse de « l'utilité » des pages web (page ranking)
- Le moteur d'interrogation
- Le module de classement

Introduction

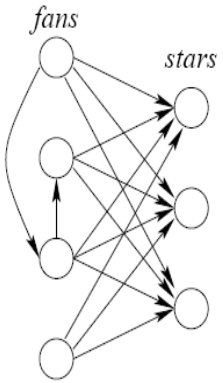
- Ce cours aborde plus ou moins en profondeur chacun des composants cités et couvre l'ensemble des notions nécessaires à la réalisation d'un moteur de recherche
  - relativement avancé d'un point de vue des fonctionnalités potentiellement couvertes
  - mais limité du point technologique et de l'implémentation de ces fonctionnalités

La structure du web



La structure du web

- Le WEB
  - Le web peut-être considéré comme un graphe orienté
    - Les sommets sont des pages HTML ou des «terminaux » (documents non HTML téléchargeables par exemple)
    - Les arcs sont des « hyper liens ».
  - La taille du web (indexable) est estimée (en 2007) à 30 milliards (30 10^9) pages
  - *(Nombre moyen de neurones dans un cerveau humain = 100 milliards)*

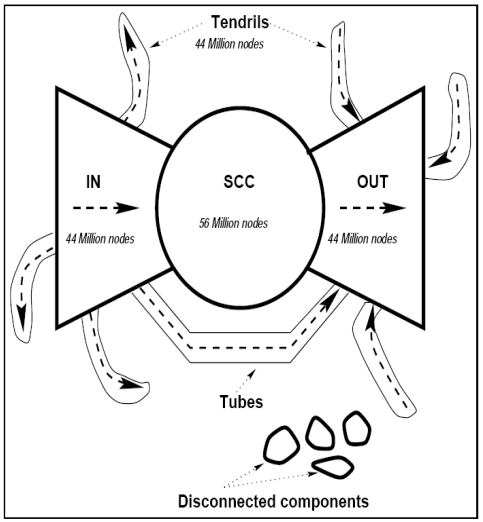


Structure classique
Que l'on retrouve sur le web
« Hub » et « Authorities »

La structure du web

The Indexed Web contains at least 5.51 billion pages (Monday, 14 September, 2020).

Source :
https://www.worldwidewebsize.com/

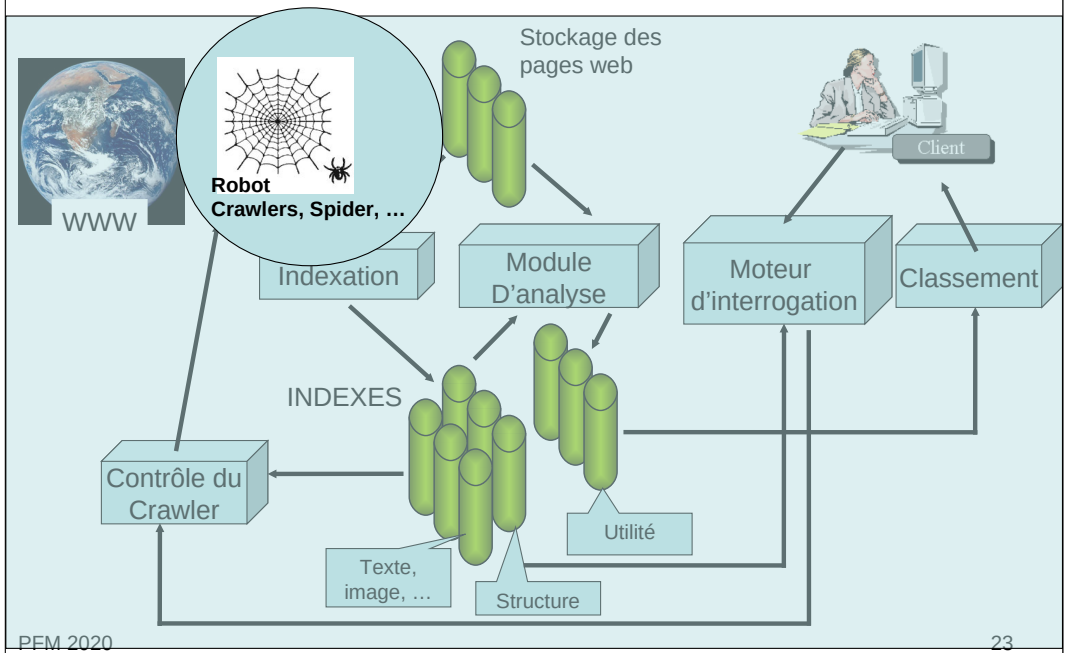


D'après : The Web as a graph - Ravi Kumar Prabhakar Raghavan Sridhar Rajagopalan D. Sivakumar ***Andrew Tomkins*** *Eli Upfaly****, 1999***

Sommaire du CM

- Introduction : Principes et Architecture d'un Moteur de Recherche sur le web
- Les robots de collecte d'information (spider, crawler)
- Le stockage des pages web
- Quelques modèles d'indexation
- Analyse de liens et évaluation de l'importance des pages
- Quelques modèles de recherche d'information
- Classification des documents
- Évaluation des performances
- La recherche de documents non textuels
  - Les séries temporelles
  - Les images
- Le WEB Invisible

Les crawlers



Les robots de collecte (crawler, spider)

- Les « crawlers »
  - parcourent la toile (récursivement) en suivant les «hyperliens »
  - Ils extraient l'information contenue dans les pages pour une analyse ultérieure

Crawler : algorithme de parcours du WEB

- Initialisation
  - S0: ensemble d'URLs à télécharger : tous les éléments de S0 sont placés dans une file d'attente Q ordonnée suivant un ordre de priorité (qu'il convient de spécifier)
- Itération
  - Prendre le 1er élément de Q
  - Télécharger la page correspondante
  - Extraire de la page téléchargée les nouvelles URLs que celle-ci contient
  - Insérer ces nouvelles URLs dans Q en respectant l'ordre défini
- Conditions d'arrêt
  - Q est vide,
  - La limite mémoire disque pour stocker les URL est atteinte
  - Le nombre d'URL max à télécharger est atteint, etc.

Crawler : quelques questions délicates

- Etant donnés la taille énorme du WEB et son taux d'évolution rapide, les questions suivantes (entre autres) peuvent être abordées
- 1. Quelles pages doivent être téléchargées ?
  - Une petite (infime) partie du web peut être indexée : comment choisit-on les pages à indexer et à rafraîchir pour maximiser l'utilité globale du moteur de recherche d'information
- 2. Comment rafraîchir une page indexée ?
  - Une fois que le crawler a indexé un nombre important de pages, celui-ci doit commencer à re-visiter les pages indexées pour le cas échéant tenir compte de leur mise à jour.
  - Bien choisir les pages à re-visiter et celles à ne pas re-visiter conditionne la « fraîcheur informationnelle» du moteur de recherche d'information : il peut être judicieux de tenir compte des statistiques d'évolution des pages indexées pour établir une heuristique de re-visite des pages «optimale »

Crawler : quelques questions délicates

- 3. Comment peut-on réduire la charge du site visité ?
  - Quand le crawler collecte une page WEB, il consomme des ressources appartenant à d'autres organisations : lorsqu'un crawler télécharge une page *p* sur un site *S*, le site *S* doit extraire la page *p* de son système de gestion de fichier : il consomme donc des ressources disque, CPU, et réseau (bande passante). Le crawler doit minimiser son impact sur ces ressources, à défaut les organisations visitées ou intermédiaires peuvent porter plainte ou limiter les accès à leurs ressources.

Crawler : quelques questions délicates

- 4. Comment peut-on paralléliser la visite et la collecte des pages ?
  - Du fait de la taille énorme du WEB, il est courant d'exploiter plusieurs crawlers fonctionnant en parallèle (nécessaire pour télécharger un grand nombre de pages en un minimum de temps).
  - Ces crawlers doivent être co-ordonnés pour éviter qu'un site soit visité plusieurs fois. Cette coordination implique une augmentation importante des communications entre les processus (et processeur) et des accès concurrents à la mémoire partagée : cette augmentation peut limiter le nombre total de crawlers à exploiter.

Crawler : quelques questions délicates

- Dans le cadre de ce cours, nous abordons uniquement les 2 premiers points, i.e. :
  - La stratégie de sélection des pages
  - La stratégie de rafraîchissement des pages
- En pratique, les architectures multi-cœurs et multi-threadées des CPU permettent de répondre en partie à la question N°4.

La stratégie de sélection des pages

- Etant donnée une page WEB, on peut définir son importance de trois manières distinctes (qui peuvent être combinées):
  - Importance guidée par l'intérêt (focus driven)
  - Importance guidée par la popularité (popularity driven)
  - Importance guidée par la position (location driven)

La stratégie de sélection des pages : Importance guidée par l'intérêt

- Le but recherché est d'obtenir des pages dignes d'intérêt pour un utilisateur ou un groupe d'utilisateurs
  - Les pages intéressantes sont celles qui répondent aux attentes des utilisateurs
  - Pour définir cette notion, on introduit le concept intermédiaire de «requête » ou de « thème » Q:
    - Etant donnée une requête *Q*, l'importance d'une page *P* est définie sous la forme d'une mesure de similarité entre la requête *Q* et la page *P*. Par exemple, cette similarité peut-être basée sur le modèle vectoriel que nous aborderons plus tard dans ce cours (parmi d'autres modèles possibles)
  - On note *IS(P)* l'importance guidée par l'intérêt
  - Il s'agit donc d'une notion relative (à une spécification de recherche) d'importance

La stratégie de sélection des pages : Importance guidée par la popularité

- L'importance d'une page dépend de sa «popularité »
  - Une manière d'estimer la popularité d'une page *P* est d'évaluer le «backlink» de *P*, i.e. le nombre de liens externes à *P* qui conduisent à la page *P*.
  - Intuitivement, une page très référencée (par des liens externes) sur la toile est plus importante qu'une page peu ou pas référencée.
  - On note *IB(P)* l'importance guidée par la popularité
  - Remarque: un crawler estime et affine *IB(P)* à partir du comptage des « backlinks » sur les pages visitées : *IB'(P)* : (cette estimation est très peu fiable en début de parcours du web et s'améliore au fur et à mesure).
  - On verra une métrique similaire (bien que plus sophistiquée) un peu plus loin dans le cours : « PageRank », noté *IR'(P)*
- Il s'agit ici d'une notion d'importance indépendante d'une spécification quelconque de recherche d'information

La stratégie de sélection des pages : Importance guidée par la position

- L'importance *IL(P)* d'une page *P* dépend de sa position (et non plus de son contenu)
  - Si une URL *u* mène à *P*, alors *IL(P)* est une fonction de *u*. Par exemple, une URL finissant par «stanford.edu» ou «mit.edu» peut être considérée comme plus intéressante qu'une URL finissant par sitepourlesnuls (ex: «http://membres.multimania.fr/sitepourlesnuls/»).
  - Une autre heuristique pénalise les URL contenant beaucoup de « / » par opposition à celles contenant peu de « / ». Par exemple, /www.univ-ubs.fr peut être considérée comme plus importante que /www.univ-ubs.fr/ufr-ssi/dpt-mis/formation/master/m2-pro/INF23xx/…
  - La notion d'importance guidée par la position peut être considérée comme un cas particulier de la notion d'importance guidée par l'intérêt. Les principes d'estimation de l'importance sont néanmoins très différents
- cette notion d'importance est également indépendante de toute spécification de recherche d'information

La stratégie de sélection des pages : combinaison des métriques

- Les trois notions précédentes peuvent être combinées pour former une « méta » métrique qu'on espère parfois plus fiable (de meilleure qualité)
  - Par exemple on peut utiliser une simple combinaison linéaire :

    $IC(P) = k_1.IS(P)+k_2.IB(P)+k_3.IL(P)$

    Avec $k_1+k_2+k_3=1$

Modèles pour les « crawlers »

- Le but étant de concevoir des crawlers qui, si possible, visitent en priorité les pages importantes, il est nécessaire de choisir une métrique pour ordonner l'importance des pages et un principe d'estimation de cette métrique (par exemple *IB(P)* et *IB'(P)*) exploitable par le crawler.
- On peut définir une notion de qualité pour ces métriques selon les modèles suivants (parmi d'autres) :
  - « Crawl et stop »
  - « Crawl et stop » avec seuil

Modèles pour les « crawlers » modèle « Crawl et stop »

- Le crawler *C* démarre sur la page initiale P_0 et s'arrête après avoir visité *K* pages (*K* est déterminé par le nombre de pages que le crawler est capable de télécharger en un seul « crawl »).
- Un crawler parfait visiterait en un « crawl » les pages $R_1, R_2,\ldots,R_k$, avec R_1 la page ayant la plus grande importance (sur tout le web), R_2 la page ayant la 2ème plus grande importance, etc.
- $R_1, R_2,\ldots R_k$ sont appelées les «hits» ou les «hot pages»

Modèles pour les « crawlers »
modèle « Crawl et stop »

- Un crawler réel visite *M* « hits » ayant une importance supérieure ou égale à R_k, avec $M \leq K$
- *Remarque : pour connaître la valeur de M, il faut connaître l'importance exacte de toutes les pages du web, ce qui est impossible. On évalue donc en général les crawler sur une petite partie (infime) du web.*
- On définit la performance du crawler C par la quantité : $P_{CS}(C) = \frac{M.100}{K}$ exprimé en %

Modèles pour les « crawlers »
modèle « Crawl et stop » avec seuil

- Nous supposons ici également que le crawler *C* visite *H* pages, mais une valeur d'importance I_s est spécifiée : le seuil d'importance
- Toute page visitée ayant une importance supérieure ou égale à I_s est prise en compte, rejetée dans le cas contraire.
- Soit *H* le nombre total de « hits » sur le web (nombre de pages ayant une importance supérieure ou égale à I_s).
- Si nous supposons (comme précédemment) que nous connaissons l'importance de toutes les pages du web, nous pouvons évaluer *H*.
- La performance d'un crawler *C* est notée : $P_{css}(C)$. C'est le % de « hits » ayant été visités par le crawler *C* lorsqu'il s'arrête.
- Un crawler *C* ayant extrait *M* hits aura pour performance :

$$P_{css}(C) = \frac{M.100}{H}$$ exprimé en %

Modèles pour les « crawlers »
Métrique pour l'ordre d'importance des pages

- Les crawlers sélectionnent les pages à traiter en fonction d'un ordre de priorité défini par une métrique : *le crawler sélectionne dans la liste des urls à traiter l'url u qui maximise la valeur d'ordonnancement associée à la métrique*
- D'un point de vue pratique, on ne peut considérer que les métriques qui exploitent uniquement l'information déjà découverte et éventuellement archivée par les crawlers (historique du parcours du crawler)

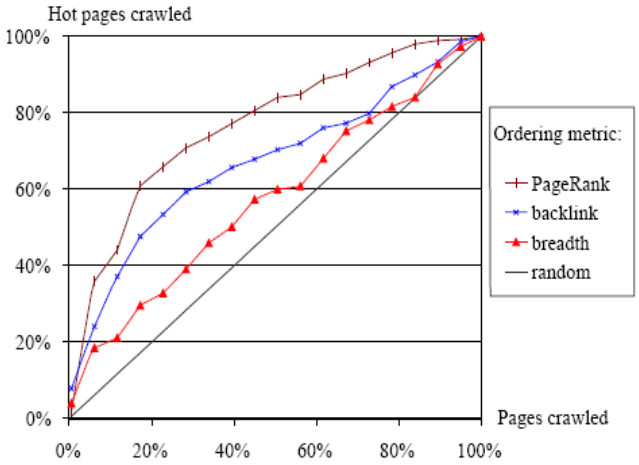
Modèles pour les « crawlers »
Métrique pour l'ordre d'importance des pages

- On a vu que la métrique *IB(P)* n'est pas exploitable, cependant elle peut être approchée par la métrique *IB'(P)*
- Une autre métrique *IR'(P)* basée sur le concept du «Page Rank » est également exploitable
- La métrique *IL(P)* basée sur l'emplacement de la page *P* (début ou fin d'URL) ne pose pas de problème d'estimation : elle est directement calculable à partir de l'url de la page.
- Pour les métriques basées sur une similarité, c'est plus délicat : certaines approches cherchent à exploiter le texte servant d'ancrage aux URL conduisant à la page P pour prédire la similarité de *P* vis-à-vis d'un critère de recherche *Q par exemple*.

Etude de cas

- On montre ici expérimentalement qu'il est possible de télécharger les pages importantes, au sens de la métrique considérée, bien avant (en moyenne) le téléchargement des pages non importantes en choisissant une métrique appropriée : projet WEBBASE (http://www9.org/w9cdrom/296/296.html) du dpt Computer Science de Stanford
  - i) définition et constitution d'un web gérable webG: les 225 000 pages extraites du site Stanford University (http://www.stanford.edu/) : on suppose que toutes les pages en dehors des 250 000 extraites ont une importance nulle
  - ii) choix d'une métrique pour la mesure d'importance : *IB(P)*
  - iii) choix du modèle de crawler : « crawl et stop » avec seuil *G = 100* : seules les pages avec un « backlink » supérieur ou égal à *100* sont considérées comme des « hits ».
  - iv) on mesure le nombre de « hits » extraits par le crawler lorsque celui-ci a extrait x% des pages du webG.

Etude de cas



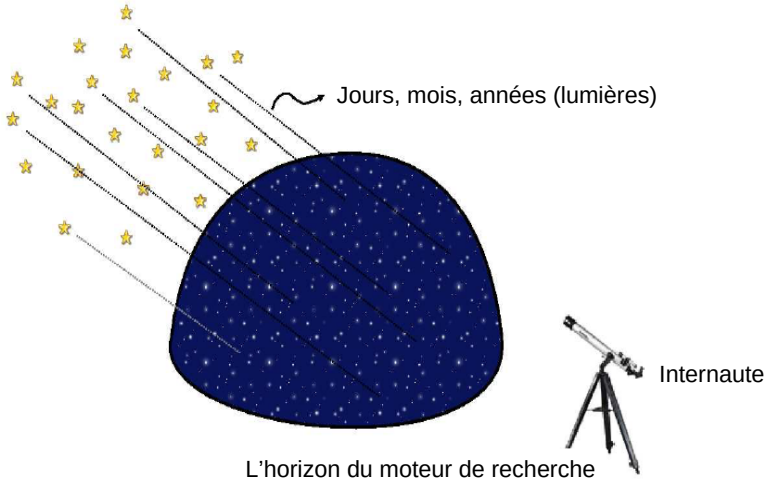
Abscisse : % de pages extraites
Ordonnée : % de hits extraits

Expérimentation pour 4 heuristiques : IR'(P) (PageRank), IB'(P) (backlink),
parcours déterministe en largeur d'abord (breadth)
aléatoire (random)

Etude de cas

- L'étude de cas montre qu'une heuristique appropriée peut améliorer significativement la performance d'un crawler, pour l'interprétation donnée à la notion de « hit »
- Remarque : l'heuristique *PageRank* est meilleure que l'heuristique *IB'(P),* bien que la métrique visée par *IB'(P)* est *IB(P).* Nous reviendrons en détail sur l'heuristique *PageRank*

La stratégie de rafraîchissement des pages



Les évènements observés n'ont pas eu lieu au même instant

La stratégie de rafraîchissement des pages

- Une fois que le crawler a sélectionné et téléchargé les pages importantes, il doit périodiquement les re-visiter et éventuellement les télécharger à nouveau pour maintenir les «hits » à jours et supprimer les pages devenues obsolètes
- Il existe un grand nombre de politiques de rafraîchissement. Nous en citons 2 :
  - Rafraîchissement uniforme
  - Rafraîchissement proportionnel

La stratégie de rafraîchissement des pages

- Pour la politique de rafraîchissement uniforme, les pages sont revisitées à la même fréquence *f*, indépendamment de leur dynamique de changement.
- Pour la politique de rafraîchissement proportionnel, le crawler revisite une page proportionnellement à la vitesse de changement de cette page :
  - Si λ_i est la fréquence de changement de la page P_i, et si f_i est la fréquence de re-visite du crawler pour la page P_i, alors, le rapport λ_i / f_i est constant (indépendant de i).
  - Cette politique implique que le crawler doit récupérer d'une manière ou d'une autre l'historique des changements pour les pages téléchargées.
- Entre ces deux politiques, quelle est celle qui conduit à la meilleure «fraîcheur » des pages téléchargées ?

Métrique pour évaluer la fraîcheur des pages téléchargées

- Intuitivement, on considère qu'un ensemble de pages est « plus frais » s'il contient plus de pages à jour :
  - Soient deux ensembles *A* et *B* contenant les mêmes *N* pages. On considère que *A* (Resp. *B*), en moyenne, maintient à jour N_A pages (Resp. N_B).
    - Si $N_A \leq N_B$, alors on considèrera que *B* est plus frais que *A*.
    - On introduit également une notion d'ancienneté: même si pour *A* et *B* toutes les pages sont obsolètes, on considèrera que *B* est plus à jour que A si A a été rafraîchi antérieurement à *B*.
    - Dans la suite, on appellera « pages réelles » les pages du web et par « pages locales » les pages téléchargées par le crawler.

Notion de « fraîcheur » des pages web

- Soit $S=\{p_1,p_2, \ldots,p_N\}$ une collection locale de *N* pages. On définit la fraîcheur de cette collection de la manière suivante :

  (*Par « à jour » on entend que la page locale est identique à la page réelle*)

$$F(p_i,t)=\begin{cases} 1 \text{ si } p_i \text{ est à jour} \\ 0 \text{ dans le cas contraire} \end{cases}$$

- Ainsi, la fraîcheur de la collection locale *S* est :

$$F(S,t)=\frac{1}{N}\sum_{i}^{N} F(p_i,t)$$

Notion de « d'ancienneté » des pages web

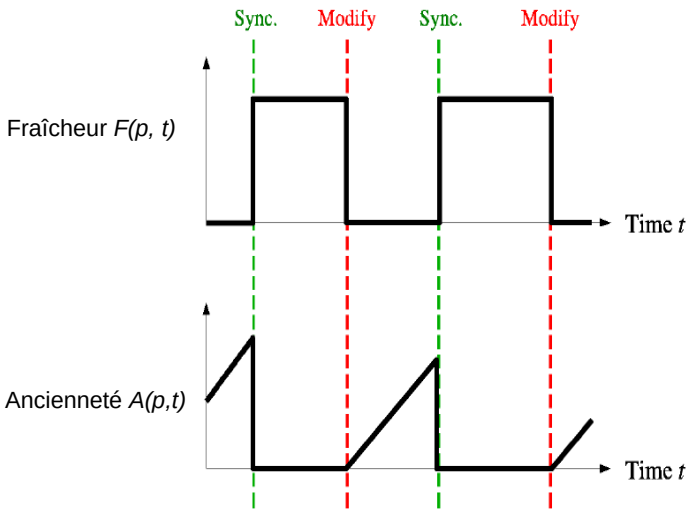
- Pour estimer l'age d'une collection, on peut définir la métrique suivante :

$$A(p_i,t)=\begin{cases} 0 \text{ si } p_i \text{ est à jour} \\ t - \text{date de dernière mise à jour} \end{cases}$$

- L'ancienneté de la collection S est alors définie par la quantité :

$$A(S,t)=\frac{1}{N}\sum_{i}^{N} A(p_i,t)$$

Ancienneté et Fraîcheur



Fraîcheur et ancienneté moyennes

- On conçoit que les grandeurs précédentes varient d'heure en heure ou de jour en jour. Pour « filtrer » les fluctuations non significatives, on va considérer les moyennes de ces grandeurs :
  - Moyennes temporelles de la fraîcheur

$$\lim_{t\to\infty}\frac{1}{t}\int F(p_i,t)\,dt \qquad \lim_{t\to\infty}\frac{1}{t}\int F(S,t)\,dt$$

  - Moyennes temporelles de l'ancienneté

$$\lim_{t\to\infty}\frac{1}{t}\int A(p_i,t)\,dt \qquad \lim_{t\to\infty}\frac{1}{t}\int A(S,t)\,dt$$

Stratégie de rafraîchissement

- La probabilité pour qu'une page *P* soit à jour à l'instant *t*, notée $u_p(t)$ décroît avec *t* si la page n'est pas revisitée
- On choisit en général un modèle de poisson pour modéliser les changements des pages web **:**
  - La date d'arrivée d'un évènement ζ suit une distribution de poisson de paramètre λ_p lorsque :
    - Prob(ζ≤t) = 1 – exp(- λ_p t), λ_p >0
    - Densité f(t) = λ_p .exp(- λ_p t)
  - Pour ce modèle, si l'intervalle de temps *t* s'est écoulé depuis la dernière mise à jour de la page *P* alors la probabilité pour que *P* soit encore à jour est :
    - $u_p(t)$ = 1- Prob(ζ≤t) = exp(- λ_p t), λ_p >0

Stratégie de rafraîchissement - estimation de λ_p -

- Le paramètre λ_p caractérise la vitesse de changement de *P.* Il peut être estimé à partir des observations *passées*, surtout si le serveur web fournit la date de dernière modification de la page lorsque celle-ci est visitée.
- Une estimation de λ_p a été proposée par Cho et Garcia-Molina :

$$\lambda_p \approx \frac{(X_p - 1) - \frac{X_p}{N_p \log(1 - X_p/N_p)}}{S_p . T_p}$$

Où :
- N_p est le nombre de visite de *P*
- X_p est le nombre de fois que le serveur a informé le crawler d'un changement pour *P*
- S_p est le temps écoulé depuis la 1ère visite pour la page *P*
- T_p est le temps total cumulé pour les périodes n'ayant pas donné lieu à modification de *P*

Stratégie de rafraîchissement - estimation de λ_p -

- Si le serveur n'indique pas la date de dernière mise à jour, on peut toujours tester les modifications potentielles en comparant la page visitée avec la page locale : X_p mesure alors le nombre de fois qu'une modification a été détectée par le crawler pour la page P.
- Dans ce cas, les auteurs proposent :

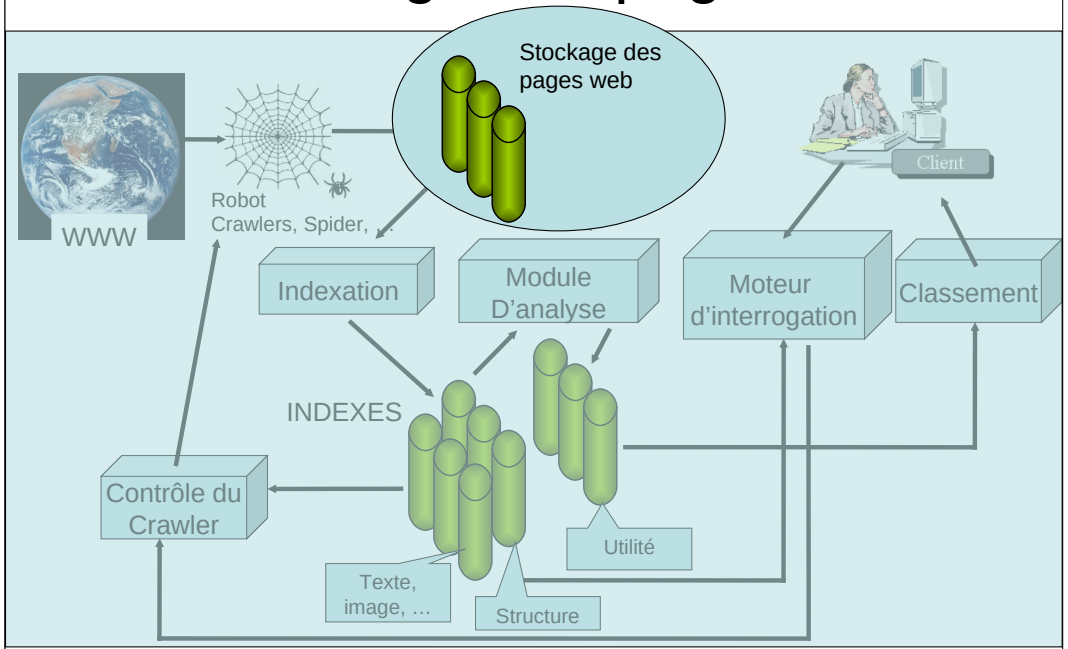
$$\lambda_p \approx \frac{- N_p \log(1 - X_p / N_p)}{S_p .}$$

Note : si la page change à chaque visite, X_p=N_p et l'estimateur n'est pas calculable (λ_p tend vers l'infini quand X_p tend vers N_p)

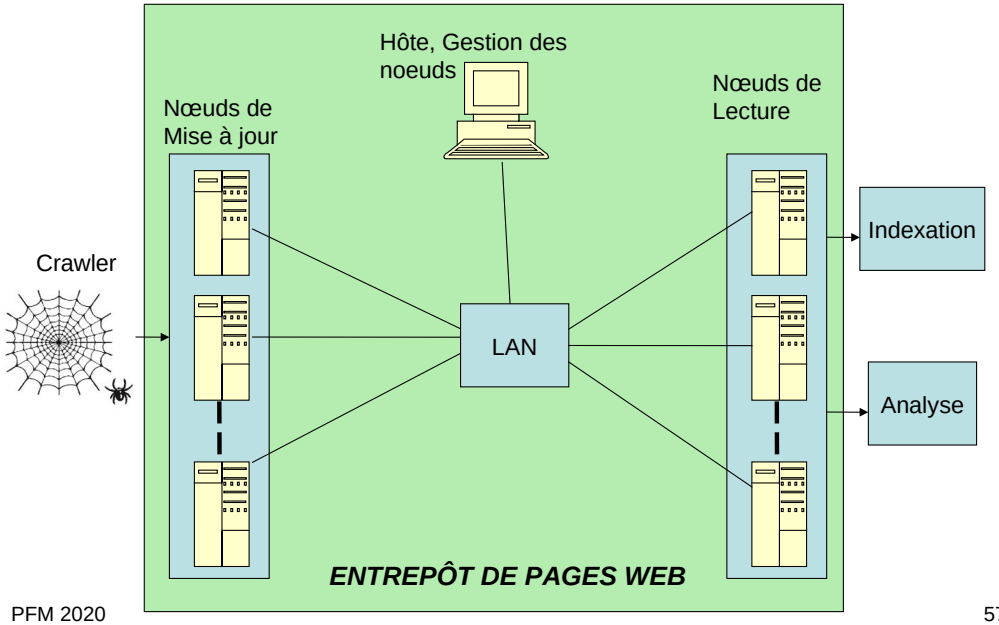
Le stockage des pages web

- Introduction : Principes et Architecture d'un Moteur de Recherche sur le web
- Les robots de collecte d'information (spider, crawler)
- *Le stockage des pages web*
- Quelques modèles d'indexation
- Analyse de liens et évaluation de l'importance des pages
- Quelques modèles de recherche d'information
- Classification non supervisée des documents
- Evaluation des performances
- Le WEB Invisible
- La recherche de documents non textuels
- Retour sur la structure de graphe du WEB

Le stockage des pages web



Le stockage des pages web - architecture générale -



Le stockage des pages web

- Le module de stockage des pages web est un **système d'entrepôt de données** permettant d'archiver des grandes collections de pages web en gérant le passage à l'échelle. Ce système doit réaliser **deux fonctions de base** :
  - Fournir une interface au crawler pour stocker les pages web
  - Fournir une API d'accès en lecture efficace pour les modules d'indexation, filtrage, recherche et d'analyse.

Le stockage des pages web - les défis posés -

- Le système d'entrepôt doit gérer une grande masse d'objets (de pages web) : dans ce sens il est d'un point de vue conceptuel similaire à un système de gestion de bases de données.
- Cependant le système d'entrepôt ne couvre que peu de fonctionnalités comparativement à un SGBD : il n'a pas besoin de couvrir les fonctions telles que la gestion des transactions, des logs, de la structure des répertoires.
- Quels sont donc les défis posés à ce type de système de stockage ?

Le stockage des pages web - les défis posés -

- **Le passage à l'échelle (scalability)**
- Le système doit de manière transparente être capable de **distribuer l'entrepôt sur un « cluster »** de machines et de disques pour tenir compte de la taille du web.

Le stockage des pages web - les défis posés -

- **Le double mode d'accès :** le système de gestion de l'entrepôt doit permettre deux modes distincts d'accès de manière très efficace :
  - **L'accès aléatoire** *(Random access)* utilisé pour extraire très rapidement une page web spécifique, étant donnée son identifiant unique *Pid*. Ce mode est exploité par le serveur de requêtes pour délivrer les copies locales des pages web (*cache copies*)
  - **L'accès séquentiel** *(Streaming access)* utilisé pour déposer la collection entière (ou un sous ensemble significatif de données) sous la forme d'un flux continu de pages web. Ce mode est exploité par le moteur d'indexation et le module d'analyse pour traiter des grands volumes de données en séquence.

Le stockage des pages web - les défis posés -

- **Mises à jour de grands volumes :** Le web changeant très rapidement, le système de gestion de l'entrepôt doit gérer des taux de mises à jours très élevés.
  - Quand les nouvelles versions des pages web sont reçues par le « crawler », l'espace occupé par les vieilles versions doit être récupéré via la réorganisation et le compactage global de l'espace de stockage.
  - D'autre part, les conflits potentiels intempestifs entre le processus de mise à jour et les applications devant accéder en lecture aux pages (recherche, filtrage, indexation, analyse, etc.) doivent être minimisés.

Le stockage des pages web - les défis posés -

- **Les pages obsolètes :** dans la plupart des systèmes de gestion de fichiers, les objets sont explicitement détruits lorsque qu'ils ne sont plus requis.
  - Malheureusement, lorsqu'une page web disparaît d'un site sur le web, l'entrepôt local n'est pas notifié de cette suppression. Le système de gestion de l'entrepôt doit donc intégrer un mécanisme pour détecter et supprimer les pages obsolètes.

Détermination du Pid

- Puisqu'une page web est l'unité logique fondamentale gérée par l'entrepôt, il est important de disposer d'un mécanisme bien défini et utilisable par tous les modules de traitement pour référencer de manière unique une page web. Ce mécanisme produit un Pid unique.
- Dans le système WebBase de Stanford par exemple, cet identificateur unique est construit en calculant une signature (par exemple le checksum ou le test de redondance cyclique) de l'URL de la page source.
- Comme une URL peut avoir plusieurs représentations (par exemple *http://www.univ-ubs.fr:80/* et *http://www.univ-ubs.fr),* une même page pourrait avoir plusieurs signatures, donc plusieurs Pid. Pour éviter ce problème, il est nécessaire de transformer les URLs sources sous une forme normalisée.
  - **Normalisation:** Une URL est normalisée par l'algorithme suivant :
    - Suppression du préfix spécifiant le protocole (*http://*) si présent
    - Suppression des spécifications du numéro de port (*:80)* si présent
    - Conversion du nom du serveur en lettres minuscules
    - Suppression de tous les slashes ("/") en fin d'URL
  - **Génération de la signature** (du Pid): une clé (entier) codée sur 64 ou 128 bits est engendrée par le biais d'une fonction de hashage.
- Note : l'utilisation d'une fonction de hashage implique une probabilité non nulle de collision (2 URLs distinctes sont associées à une même signature par la fonction de hashage). Plus la taille de la signature (clé) est grande, plus la probabilité de collision est faible :
  - Pour 100 millions de pages en entrée et une signature sur 64 bits, la probabilité de collision est de 3.10^{-4}.
  - Pour 10 milliards de pages et une clé sur 128 bits, la probabilité de collision est de 10^{-48}.

Conception d'un entrepôt distribué

- On considère un entrepôt générique capable de fonctionner sur un « cluster » de nœuds de stockage interconnectés. Trois points clés affectant les caractéristiques et les performances de l'entrepôt doivent être considérés :
  - *La distribution des pages sur les nœuds*
  - *L'organisation physique des pages sur un nœud*
  - *La stratégie de mise à jour*

Conception d'un entrepôt distribué
- *politique de distribution* -

- Les pages web peuvent être distribuées sur les nœuds selon différentes politiques :
  - Par exemple, avec une politique de distribution *uniforme,* tous les nœuds sont traités de manière identique ; une page donnée peut être affectée à n'importe quel nœud du cluster, indépendamment de son *Pid*. Les nœuds stockent des portions de la collection, proportionnellement à leur capacité de stockage.
  - Par opposition, une politique de distribution par « hashcoding » affectera les pages aux nœuds en fonction de leur Pid (Le Pid, signature de l'URL traitée, est utilisé pour décider du nœud d'affectation) : on associe à chaque nœud de stockage un domaine continu de valeurs ; chacun des nœuds contient les pages dont le Pid appartient au domaine de valeurs qui lui est associé.

Conception d'un entrepôt distribué
- *L'organisation physique des pages* -

- Sur chaque nœud, trois opérations peuvent être exécutées :
  - *addition/insertion d'une page*,
  - *L'accès séquentiel rapide (high-speed streaming)*,
  - *Et l'accès aléatoire aux pages* (*random page access)*.
- L'organisation physiques des pages est un facteur clé qui détermine l'efficacité avec laquelle un nœud exécute ces trois opérations.

Conception d'un entrepôt distribué
- *L'organisation physique des pages* -

Il y a plusieurs options pour l'organisation des pages :

- Approche par partitionnement des disques
- Approche par fichier de log
- Approche hybride : hash-log

Conception d'un entrepôt distribué
- L'organisation physique des pages -

- **Approche par partitionnement des disques**
  - Par exemple, une organisation basée sur un partitionnement des disques (en « *buckets* ») exploite une table de hashage pour associer chaque page à un élément de la partition physique en fonction du *Pid* de la page. L'intérêt est de jouer sur la taille des éléments de la partition afin que ceux-ci puissent rentrer en mémoire RAM dans le but d'accélérer les traitements.

Conception d'un entrepôt distribué
- L'organisation physique des pages -

- **Approche par fichier log**
  - Puisque l'ajout de nouvelles pages est une opération fréquente, une structure type fichier « log » (journal) peut être avantageuse. Dans ce cas, le disque dans son ensemble est vu comme un grand fichier log continu, dans lequel, les nouvelles pages sont ajoutées en fin de fichier (« append ») . Pour assurer la fonction « d'accès aléatoire », on utilise en général une structure d'arbre binaire (*B-Tree*) qui associe les Pids aux adresses physiques de la page dans le fichier de log.

Conception d'un entrepôt distribué
- L'organisation physique des pages -

- **Approche hybride : hash-log**
  - On peut également concevoir une approche hybride « *hash-log* », pour laquelle le disque est divisé en grandes plages séquentielles de mémoire (« extents ») qui rentrent en mémoire *RAM*. Les pages sont associées aux plages de mémoire via une fonction de hashcode et chaque plage mémoire est organisée comme un fichier de log.

Conception d'un entrepôt distribué
- L'organisation physique des pages -

- Analyse de performance des solutions décrites :

	Fichier-log + BTree	***Partitionnement + Hash***	***Mixte (Hash-log)***
Accès séquentiel	++	-	+
Accès aléatoire	+-	++	+-
Ajout de page web	++	-	+

D'après « WebBase : A repository of web pages

Jun Hirai[1, 3] Sriram Raghavan[2] Hector Garcia-Molina[2] Andreas Paepcke[2]
1System Integration Technology Center, Toshiba Corp., 3-22 Katamachi, Fuchu, Tokyo 183-8512, Japan
2Computer Science Department, Stanford University, Stanford, CA 94305, USA
jun.hirai@toshiba.co.jp, {rsram, hector, paepcke}@db.stanford.edu

Conception d'un entrepôt distribué
- *Stratégie de mise à jour* -

- Puisque les mises à jour sont sollicitées par le crawler, la conception de la fonction de mise à jour dépend des caractéristiques d'exploitation du crawler. En particulier, on peut spécifier cette exploitation selon trois axes :
  - Le mode : « Batch » vs. Continu
  - L'étendue du parcours de re-visite : Partielle vs. Totale
  - Mise à jour en lieu et place vs. mode « shadowing »

Conception d'un entrepôt distribué
- *Stratégie de mise à jour* –
- mode batch ou continu -

- Un crawler fonctionnant en mode *«Batch»* est exécuté périodiquement (i.e. une fois par mois) Le crawler fonctionne soit sur un intervalle de temps de durée fixe (inférieure à la période choisie) soit il s'arrête lorsqu'un nombre fixe de pages ont été re-visitées. La mise à jour n'est active que pendant la période d'activation du crawler.
- Par opposition, un crawler fonctionnant en mode «continu» fonctionne sans pause, en soumettant à l'entrepôt de manière continue des mises à jour ou de nouvelles pages.

Conception d'un entrepôt distribué
- *Stratégie de mise à jour* –
- étendue partielle ou totale –

- Un crawler peut travailler en étendue totale ou partielle. Dans le premier cas, les pages issues du nouveau parcours de la toile remplacent intégralement les anciennes pages existantes dans l'entrepôt). Dans le second cas, la nouvelle collection de pages web est créée à partir des mises à jour conditionnées par la re-visite partielle des pages de la collection existante.

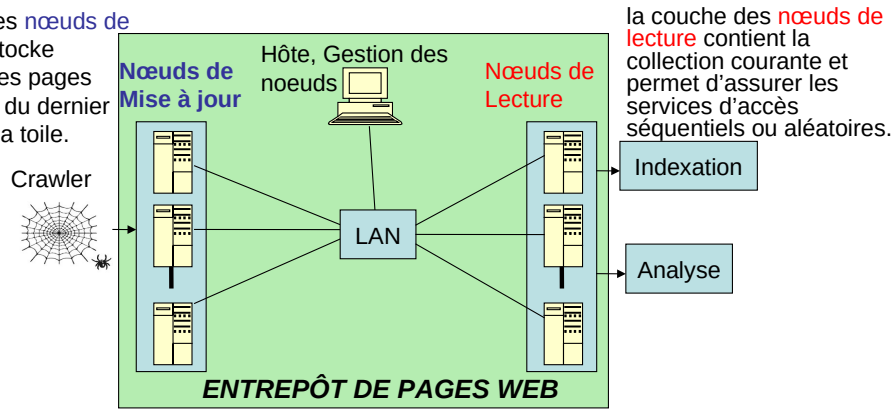
Note: cette distinction s'applique que pour les crawler fonctionnant en mode « batch ».

Conception d'un entrepôt distribué
- *Stratégie de mise à jour* –
- lieu et place / shadowing–

- En fonction du choix de ces paramètres, le système de gestion de l'entrepôt peut soit implémenter un mécanisme de mise à jour «en lieu et place» (in place update), ou un mécanisme de «shadowing»
  - Pour le mécanisme en « lieu et place » les pages extraites par le crawler remplacent directement les pages obsolètes dans la collection existante.
  - Avec le « Shadowing », les pages extraites par le crawler sont stockées dans des fichiers temporaires, et les mises à jour ont lieu lors d'une deuxième étape séparée.

Conception d'un entrepôt distribué
- *Stratégie de mise à jour* –

La couche des nœuds de mise à jour stocke l'ensemble des pages extraites lors du dernier parcours de la toile.



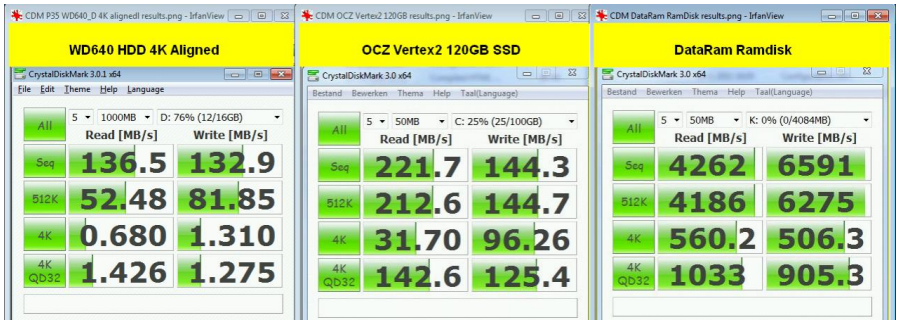
la couche des nœuds de lecture contient la collection courante et permet d'assurer les services d'accès séquentiels ou aléatoires.

- Pour le « shadowing »la séparation entre les couches de lecture et de mise à jour est complète. Un nœud de stockage n'a jamais à gérer de manière concurrentes des opérations de mise à jour (écritures) et de lecture, ce qui évite les conflits d'accès et simplifie l'architecture
- En contre partie, puisqu'il y a des délais entre l'extraction des pages par le crawler et leur disponibilité en lecture, le «shadowing» réduit la «fraicheur» moyenne de l'entrepôt.

Conception d'un entrepôt distribué
- *Stratégie de mise à jour* –

- L'expérience semble montrer que :
  - un crawler fonctionnant en mode « batch » se « marie » bien avec un mécanisme de stockage exploitant le « shadowing »
  - Un crawler fonctionnant en mode continu se marie bien avec un mécanisme de stockage exploitant le mécanisme de mise à jour « en lieu et place ».

RAM v.s. SSD v.s. HDD



Sommaire du CM

- Introduction : Principes et Architecture d'un Moteur de Recherche sur le web
- Les robots de collecte d'information (spider, crawler)
- Le stockage des pages web
- Quelques modèles d'indexation
- Analyse de liens et évaluation de l'importance des pages
- Quelques modèles de recherche d'information
- Classification des documents
- Evaluation des performances
- La recherche de documents non textuels
  - Les séries temporelles
  - Les images
- Le WEB Invisible