Architecture et Prérequis AFS

Version AFS :	6.5
Version de documentation :	6.5-02
Date :	15 / 01 / 2007
Référence :	AFS/DOC/ARCHI

Table des matières

1. Présentation de la solution

1.1. Architecture conceptuelle

1.1.1. Une solution à base d'agents logiciels

Antidot Finder Suite (AFS) est une boîte à outils d'agents[1] logiciels permettant de construire des solutions de recherche et d'accès à l'information. Pour bâtir une solution parfaitement adaptée, il suffit de choisir, d'assembler et de configurer un ou plusieurs agents parmi tout un ensemble de modules prêts à l'emploi. Chacun des agents mis en œuvre aura la charge d'une source de données (sites web, serveurs de fichiers ou de mails, archives, bases de données, annuaires, flux XML, catalogues électroniques, …).

Cette technologie à base d'agents permet de traiter au mieux chaque source de données en prenant en compte ses contraintes d'accès, de sécurité, de format, de structure et de contenu. Les agents les plus couramment utilisés sont :

- AntiBot : indexation de sites et de serveurs de fichiers ;
- IndeXML : indexation de données structurées comme les SGBD, les flux XML, RSS, … ;
- eCatalogue : catalogues électroniques ;
- Directory : annuaires ;
- Kword : associations de mots clés à des contenus éditoriaux ;
- Hit-Parade : analyse des recherches des utilisateurs, des documents les plus accédés, …

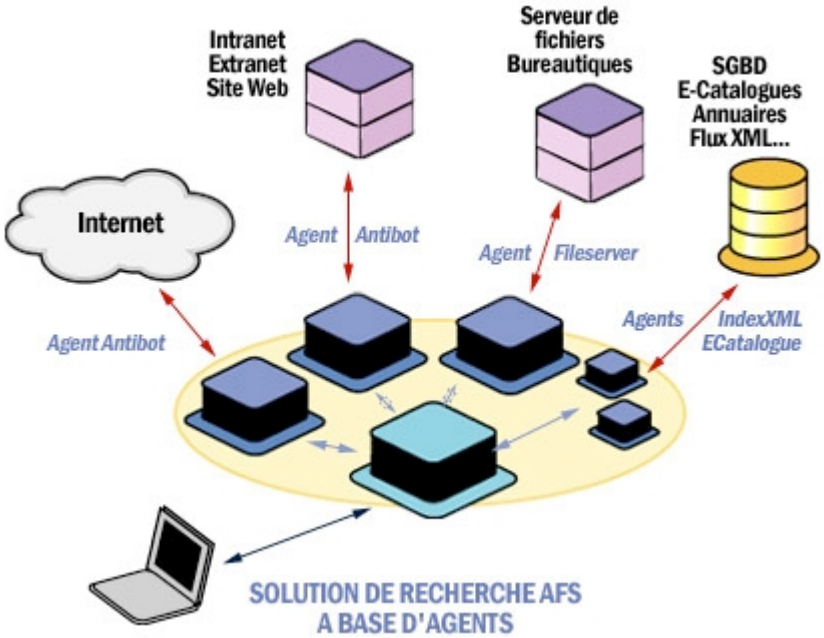

Illustration 1 : agents logiciels

1 un agent est un logiciel qui remplit de façon autonome une tâche particulière au sein d'un système plus vaste.

Lorsqu'un utilisateur pose une question, celle-ci est envoyée à l'élément central de la solution appelé **RMS** (Request Monitoring System) qui a la charge de transférer la requête aux différents agents qui composent la solution, puis de récupérer leurs réponses afin de les agréger pour produire la réponse finale qui sera fournie à l'utilisateur. C'est au niveau du RMS que sont appliquées les règles de sécurité, de tri, de dédoublonnage, de pondération et de catégorisation qui sont automatiquement configurées lors de l'installation de la solution et ajustées par la suite.

Techniquement parlant, AFS met en œuvre des mécanismes de propagation / agrégation de flux XML avec la possibilité de traiter à la volée ces flux XML en appliquant des règles métiers qui sont spécifiées de façon externe.

Cette architecture unique fait d'AFS le produit le plus à même d'aborder les nouveaux challenges de la recherche d'information en intégrant de nouveaux supports et de nouveaux formats tels que l'audio et la vidéo, mais aussi de s'interfacer avec des systèmes externes pour consolider leurs connaissances.

1.1.2. Une solution orientée objets

AFS est une solution extrêmement modulaire entièrement développée en C++ selon une architecture logicielle orientée objets très avancée : chaque fonctionnalité est implémentée par une classe dont le comportement est défini par des règles externes définies dans des fichiers de configuration au format XML. Ces classes peuvent également être surchargées pour modifier leur comportement et traiter les cas les plus complexes.

Grâce à son architecture ouverte à base de plug-ins, des modules peuvent être ajoutés afin d'implémenter et d'ajouter de nouvelles fonctionnalités au système.

Le schéma ci-dessous montre comment les différentes couches fonctionnelles s'articulent, depuis la problématique de récupération de l'information, en passant par l'analyse, l'indexation et les répondeurs. Ces différentes couches et modules sont regroupés au sein d'agents logiciels qui exposent leurs fonctionnalités à travers des web services internes et fonctionnent en grille afin d'offrir des performances et une scalabilité de tout premier ordre.

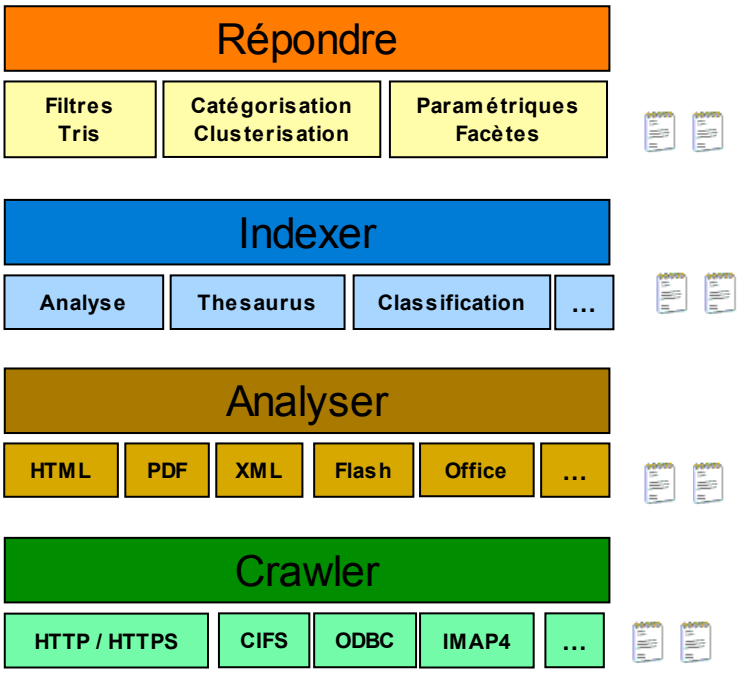

Illustration 2 : couches logiques d'AFS

1.2. Architecture logique

D'un point de vue technique, AFS se décompose en **3 blocs distincts** :
- La partie **indexation**, elle-même composée de différents modules : crawlers (accèdent aux données selon le protocole et le format adaptés), indexers (analysent les données), gestionnaires de données et métadonnées, module de surveillance et pilotage, ...

- La partie **réponse aux requêtes** composée des différents agents logiciels en charge de répondre, du module central (RMS) qui reçoit les requêtes et les transmet aux agents.

- Le **back-office** qui donne accès aux différents outils et qui a également la charge d'enregistrer les requêtes à des fins de statistiques, d'analyse et d'audit.

Définitions :
L'ensemble constitué par les modules d'indexation et le back-office est appelé **back-end AFS**.
L'ensemble constitué par les agents répondeurs et le RMS est appelé **front-end AFS**.

1.2.1. Composants fonctionnels de l'indexation

Le schéma ci-dessous illustre l'ensemble composants fonctionnels susceptibles d'être mis en oeuvre dans le cadre d'une indexation multisources en mettant en évidence leur interaction :

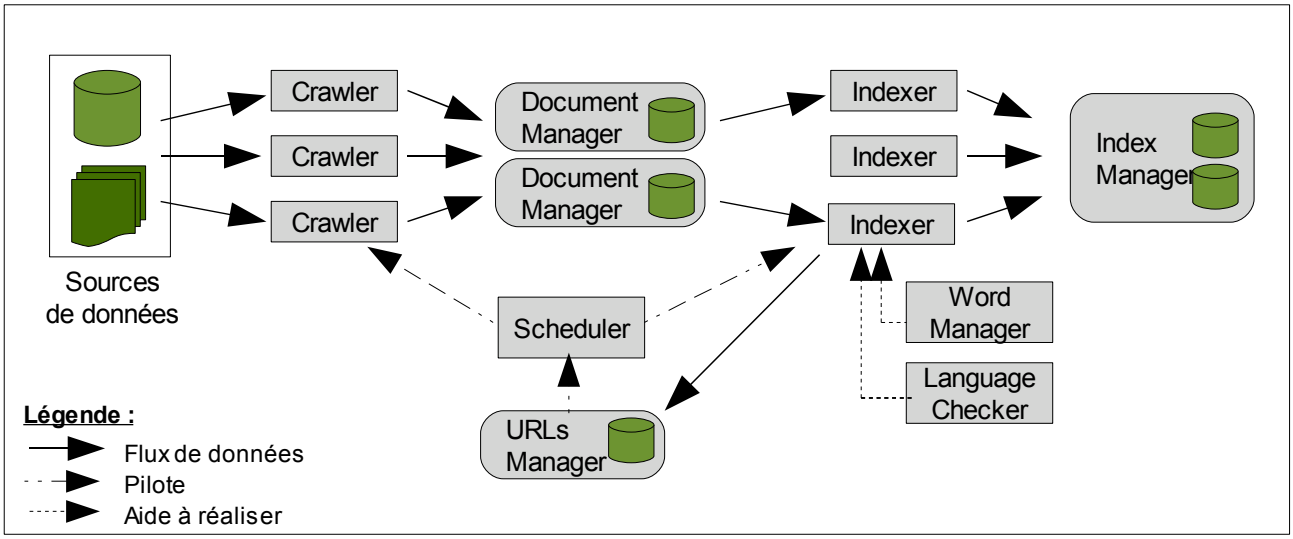

Illustration 3 : composants fonctionnels de l'indexation

Les modules impliqués dans le processus d'indexation sont les suivants :

- **le Scheduler** est en charge de lancer, d'ordonnancer et de coordonner les autres éléments selon la configuration. Il a aussi pour rôle d'affecter leurs tâches aux différents acteurs de l'indexation (liste d'URLs à télécharger, éléments à indexer, ...), agissant ainsi comme intermédiaire entre l'*URLs Manager*, et les C*rawlers* et les I*ndexers*.
- **l'URLs Manager** stocke les URLs des "documents" à crawler et à indexer. La liste des URLs peut être soit chargée à partir de la configuration, soit créée dynamiquement lors de la lecture des sources de données et lindexation de chaque élément de contenu (document HTML, mail, fichier, enregistrement de base de données, ...). Cet élément utilise pour le stockage physique une base de données relationnelle, habituellement MySQL.
- **Les Crawlers** : ces modules génériques ont la charge de récupérer les données à indexer dans les différentes sources selon les ordres transmis par le S*cheduler* et l'*URLs Manager*. Ils utilisent pour cela des plugins qui implémentent les protocoles et méthodes d'accès adaptés aux sources à indexer. C'est au niveau des crawlers que sont gérées les problématiques de sécurité d'accès aux données (authentification, gestion

des sessions, ...) et que les documents peuvent être transformés ou enrichis avec des métadonnées avant d'être envoyés aux D*ocuments Managers* pour être stockés.

- **Les Document Managers** stockent les données extraites des sources par les C*rawlers* afin de les servir à d'autres modules selon les besoins et la configuration : indexation, visualisation par les utilisateurs des documents tels qu'ils étaient lorsqu'ils ont été crawlés (fonction de cache), généreration les snippets[2] ou des résumés.
- **Les Indexers** sont les modules chargés de l'indexation des documents à proprement parlé, une fois qu'ils ont été crawlés : ils réalisent l'analyse structurelle (format) des données puis les analyses lexicales (tokenisation), syntaxiques et grammaticales des textes. Ils dialoguent pour cela avec le *Word Manager* et le *Language Checker*. Ils sont également chargés d'extraire les expressions et les concepts. Les nouvelles URLs détectées dans les documents lors de leur indexation sont extraites et envoyées à l'*URL Manager*. Le résultat de l'indexation d'un document est une liste de tuples qui est envoyée à l'*Index Manager*.
- **Le Word Manager** crée et maintient à jour la liste de tous les lexèmes[3] présents le corpus[4]. Cette liste est par exemple utilisée pour les fonctions de suggestion orthographique.
- **Le Language Checker** trouve ou vérifie les langues des documents.
- **L'Index Manager** reçoit des listes de tuples {page_id, word_id's} des Indexers, les stocke puis les utilise pour construire les index finaux qui seront utilisés par les agents répondeurs (Replyers Agents). Pour cela il réalise des opérations de type inversion d'index, calculs de scores de pertinence, ...

Module	***Binaire***	***Instance***
Scheduler	`as_init`	Un seul S*cheduler* par instance d'AFS.
URLs Manager	`as_umanager`	Plusieurs *URLs Managers* peuvent être utilisés en parallèle dans le cas d'un très grand corpus (de plusieurs dizaines ou centaines de millions de documents).
Crawlers	`as_crawl`	Il est possible de lancer autant d'instances que nécessaire afin de réaliser la récupération des données dans le temps le plus court. Un seul crawler peut générer plus de 20Mbps de bande passante. Il faut donc faire attention à ne pas surcharger la source de données.
Document Managers	`as_dmanager`	Il est possible d'utiliser autant d'instance (et de serveurs) que nécessaire selon la volumétrie et la capacité de bande passante des machines utilisées.
Indexers	`as_webindex` `as_xmlindex`	Utiliser autant d'instances que nécessaires pour indexer les documents crawlés dans le temps désiré.
Word Manager	`as_wmanager`	Une instance maître et des instances esclaves pour réduire la contention et améliorer les performances lors de l'indexation.
Language Checker	`as_lcheck`	En utiliser autant que nécessaire (jusqu'à un par *Indexer*) selon les besoins de performance.

Index Manager	`as_idmanager`
	`as_reverse`
	`as_pscore`
		Un seul *Index Manager* par instance d'AFS.

2 les snippets sont les résumés des documents utilisés lors de l'affichage dans les listes de réponses.
3 un lexème est une unité lexicale : un mot, un chiffre, une référence dans un catalogue, une adresse email, ...
4 le corpus est le nom donné à l'ensemble des documents et données considérés.

1.2.2. Composants fonctionnels pour répondre

Les différents index générés par l'*Index Manager* servent de données de travail pour les agents répondeurs qui constituent la solution, de même que les snippets extraits seront utilisés pour constituer les résumés des réponses.

Les différents **agents de réponse** sont assemblés afin de constituer un arbre dans lequel interviennent :

- un **agent racine**, appelé *Root Node*, qui fait partie intégrante du Request Manager ;
- des **agents intermédiaires**, appelés *Cell Nodes ;*
- des **agents répondeurs**, appelés *Replyer Agents*, qui travaillent avec les index afin de répondre aux questions reçues au mieux de leur compétence.

Le schéma ci-dessous illustre l'ensemble composants fonctionnels mis en oeuvre pour répondre aux requêtes :

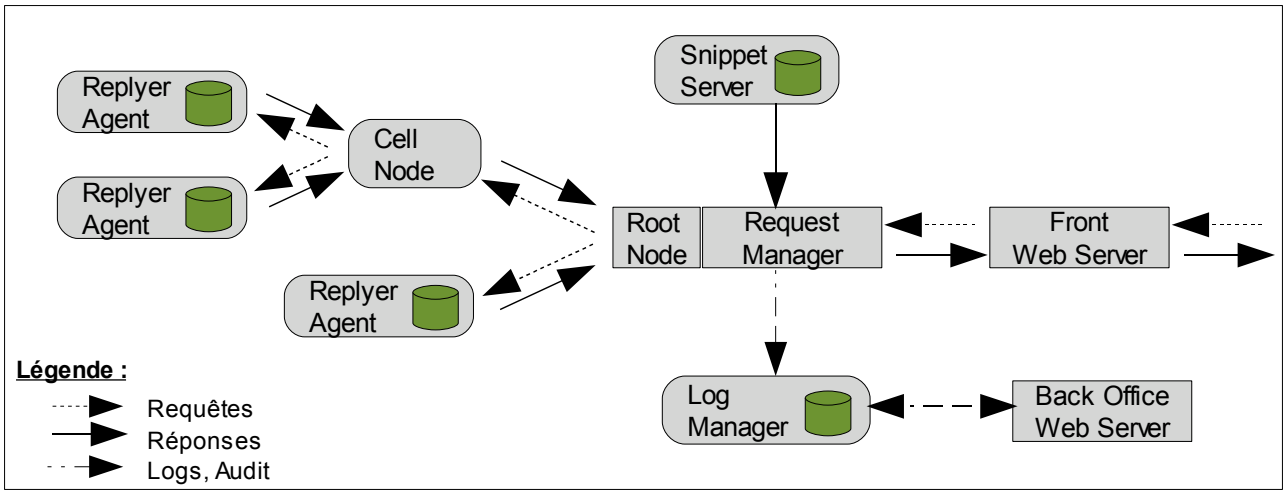

Illustration 4 : composants fonctionnels du reséau d'agent de réponses.

L'agent racine (*Root Node*) et les cellules intermédiaires (*Cell Nodes*) sont utilisés pour appliquer des traitements et des règles métier :

- lors de la propagation des requêtes vers les agents répondeurs : analyse linguistique, application de thésaurus, règles de filtrage et de sécurité ;
- lors de la récupération des réponses et de la fabrication des listes : tri, dédoublonnage, pondération, règles de sécurité, catégorisation, paramétrisation, ...

Les autres éléments qui apparaissent dans le schéma ci-dessus sont :

- le serveur web frontal (*Front Web Server*) qui reçoit les requêtes soumises en HTTP (méthode Get ou Post) soit directement par les utilisateurs soit par une autre application (serveur web, ...) ;
- le *Request Manager* qui a la charge d'identifier parmi toutes les instances des différents agents celles qui sont disponibles afin de constituer un arbre de réponse auquel la requête est transmise ;
- le *Snippet Server* qui fournit les éléments des résumés utilisés dans la liste de réponses finale ;
- le *Log Manager* auquel sont envoyées toutes les informations à enregistrer à des fins d'analyse : requêtes reçues, temps de réponse de chacun des agents, temps de

réponse final, ...

- le *Back Office* qui permet d'accéder aux logs et aux statistiques, ainsi que de configurer les différents agents (indexation et réponse).

Définitions :

Le **RMS** (Request Monitoring System) est l'ensemble composé du *Request Manager* et du *Root Node*.

Un **service de recherche** est un ensemble d'agents assemblés au sein d'une arborescence et héritant d'une configuration donnée.

1.2.3. Composants fonctionnels pour l'interrogation et l'affichage

Interrogation du service de recherche

Le service de recherche, le plus souvent accessible par un formulaire sur une page de site Web, est utilisable via le protocole HTTP. Suivant ce protocole, le client Web envoie une requête HTTP, réceptionnée par un serveur Web.

C'est par l'intermédiaire du programme CGI (Common Gateway Interface) *findall*, qui permet de faire l'interface entre le serveur Web Apache et la solution qu'AFS réceptionne les requêtes et les transmet à une arborescence d'agents désignée par le Request Manager. L'arborescence d'agents lui rend la réponse qu'il retransmet par l'intermédiaire du serveur Web.

Ce programme CGI peut être interrogé en méthode GET ou POST (selon la configuration choisie – cf. 3.7.1.3 section Agents/CGI_Front_End), et interprète un certains nombre de paramètres AFS, précisés dans le Manuel de Configuration (cf. Manuel de Configuration AFS, ch. 3).

Affichage d'une page de réponses

La réponse rendue par un service de recherche se présente sous la forme d'un flux XML. Il est donc nécessaire de transformer ce flux pour mettre en forme les informations dans un langage interprétable par les navigateurs Web (communément le (X)HTML) ou les applications qui exploitent AFS.

Cette étape de transformation est assurée par la technologie XSLT, invoquée avant d'envoyer la réponse finale.

Les informations concernant cette étape sont détaillées dans le Manuel de Configuration (cf. Manuel de Configuration AFS, ch. 5).

Contribution aux statistiques du back-office

Les liens des réponses de la page de réponses, pointent logiquement directement sur l'information désignée. Afin de contribuer à l'enrichissement des statistiques du back-office (Hit-Parade), à l'enregistrement et à l'analyse des actions des utilisateurs, ces liens peuvent désigner le programme CGI d'AFS *redirect*. Son rôle est simplement d'enregistrer le lien de la réponse cliqué par l'internaute et de le rediriger vers l'URL attendue.

Les informations concernant l'invocation de ce programme sont détaillées dans le Manuel de Configuration (cf. Manuel de Configuration AFS, ch. 3).

1.3. Architecture physique

1.3.1. Choix de l'architecture

Qu'il s'agisse d'indexer des sources ou de répondre aux requêtes, l'ensemble des modules décrits ci-avant peuvent exister un seul ou plusieurs exemplaires, sur une ou plusieurs machines afin d'assurer les performances ou la disponibilité de la solution selon les besoins et les contraintes.

Ce découpage, qui fait correspondre à chaque fonction un module spécifique capable de fonctionner de façon autonome, offre une très grande souplesse de mise en oeuvre. Ainsi toutes les configurations sont envisageables, depuis une unique machine qui héberge l'ensemble des modules et réalise toutes les fonctions, jusqu'à la mise en oeuvre de grilles de plusieurs dizaines ou centaines de serveurs permettant de réaliser l'indexation de milliards de pages web et de répondre à des centaines de millions de requêtes par mois.

Par exemple, en cas de très forte volumétrie, un index peut être découpé en plusieurs tranches et chaque morceau confié à un agent répondeur différent. Un agent intermédiaire de type *Cell Node* se charge alors de masquer se découpage au reste de la solution et de virtualiser ces agents comme un agent unique traitant l'ensemble de l'index de la source.

Les points à prendre en compte dans le choix de l'architecture physique sont :

- L'**adéquation avec une architecture existante** : il est par exemple possible de déployer les modules d'indexation dans un périmètre "sécurisé" ayant accès aux données, alors que les agents de réponse sont opérés dans une zone "publique" distincte.
- La **sécurité** : la possibilité de séparer physiquement les modules fonctionnels permet d'assurer une sécurité optimum.
- La **volumétrie** : selon le nombre de sources et de documents et selon le mode (total ou différentiel) et la fréquence d'indexation (hebdomadaire, quotidienne, en continu, ...), il est possible de multiplier le nombre de *Crawlers* et d'*Indexers*.
- Le **trafic** : selon le nombre de requêtes à servir, il est possible de mettre des ressources matérielles supplémentaires afin de multiplier le nombre d'agents répondeurs.
- La **performance** : l'ajout de ressources matérielles permet de gagner en performance, particulièrement en cas de très forte volumétrie ou de trafic important.
- La **disponibilité** : même avec des volumétries documentaires limitées et un trafic ne nécessitant pas la multiplication des ressources matérielles, il est possible de faire tourner les modules sur plusieurs serveurs, en particulier ceux de l'architecture de réponse (*RMS*, agents répondeurs, *Snippet Servers*, *Log Managers*) afin de garantir la disponibilité et la qualité du service même en cas de très forte variation de charge ou de pertes de serveurs. Cette disponibilité est gérée de façon automatique par AFS qui détecte automatiquement les ressources disponibles. Il est ainsi possible d'ajouter ou d'enlever à chaud des serveurs hébergeants des modules. Lorsqu'ils sont ajoutés, les modules s'inscrivent automatiquement auprès de la solution afin de recevoir leur part de travail. Le protocole de communication interne à AFS a été conçu pour surveiller chaque module afin que ceux qui sont surchargés, en erreur ou arrêtés ne reçoivent plus de travail.
- Le **profil des serveurs** : le profil des serveurs utilisés a un impact majeur sur les performances, les capacités de la solution (en termes de documents indexables et de capacité de réponse). La section suivante donne des indications sur les profils matériels adaptés.

Afin de maximiser l'utilisation des ressources matérielles, celles-ci peuvent bien entendu être partagées avec d'autres logiciels, mais il est également possible de programmer leur usage selon les heures de la journée. Ainsi, un serveur peut être utilisé pour réaliser l'indexation de

données durant la nuit, et se transformer en serveur hébergeant des répondeurs pendant la journée afin d'absorber la charge de trafic.

1.3.2. Exemples types d'architectures physiques

Les architectures les plus classiques que l'on peut mettre en oeuvre sont :

Type	*Composition*	*Avantages*
1-serveur	Tous les modules sur le même serveur.	⇨ Convient parfaitement aux environnements d'entreprise (intranet, ...) ⇨ Simplicité de mise en oeuvre et d'administration ⇨ Coût réduit
2-serveurs	- 1 serveur pour l'indexation et le back-office - 1 serveur pour répondre aux requêtes	⇨ Permet de séparer l'indexation et le back-office de la partie frontale qui répond aux requêtes ⇨ Sécurité accrue de la plateforme ⇨ Possibilités de trafic et de volumétrie supérieures ⇨ Facile à mettre en oeuvre dans des cas de figure type "DMZ"
3-serveurs	- 1 serveur pour l'indexation et le back-office - 2 serveurs pour répondre aux requêtes	⇨ Haute disponibilité grâce à la redondance des répondeurs ⇨ Sécurité accrue grâce à la séparation du back-end et du front-end
3-couches	- un ou plusieurs serveurs pour le front-end - un ou plusieurs serveurs pour l'indexation - un ou plusieurs serveurs pour le back-office (y compris les logs)	⇨ Sécurité optimale puisque chaque ensemble fonctionnel est séparé : depuis le front-end, il est impossible d'accéder aux données ou aux logs et au back-office. ⇨ Evolutivité maximale puisque les différentes couches sont déjà disjointes.
Grille	Répartition sur des serveurs différents des frontaux (RMS), des agents et des snippet servers.	⇨ En répartissant les différents modules sur des ressources adaptées, il est possible de contruire une véritable grille de réponse offrant performance, haute disponibilité et évolutivité, permettant ainsi d'indexer plusieurs centaines de millions de documents et de répondre à plusieurs millions de requêtes par jour.

2. Prérequis logiciel et matériel

2.1. Prérequis logiciel

La solution AFS est disponible sur Linux et sur toute plateforme Unix compatible Posix (Solaris, Aix, …). Même installée sur un de ces systèmes, elle est capable d'indexer les contenus (sites web, serveurs de fichiers, bases de données…) indépendamment de leur système d'exploitation et du logiciel.

Les distributions sur lesquelles AFS est certifié sont :

- Linux RedHat Enterprise Linux RHEL 3 et 4 (x86)
- Linux Debian Sarge et Etch (x86)

Les composants logiciels nécessaires pour faire fonctionner le back-end AFS (indexation et back-office) sont :

Logiciel	Version	Usage
Apache	1.x ou 2.x	Pour le back-office.
MySQL	version 4.1 minimum	Pour l'URLs Manager. Pour le back-office. Une même instance de MySQL peut être utilisée pour les deux composants, AFS travaillant sur deux bases différentes.
Java VM	1.5	Runtime Java permettant de faire fonctionner Tomcat. Nécessaire uniquement pour le back-office.
Tomcat	5.5.x	Nécessaire uniquement pour le back-office

Les composants logiciels nécessaires pour faire fonctionner le front-end AFS sont :

Logiciel	Version	Usage
Apache	1.x ou 2.x	Pour les frontaux AFS chargés de recevoir les requêtes.
MySQL	Version 4,1 minimum	Pour le Log Manager.

Il est laissé au lecteur le soin de se reporter aux documentations de chacun des produits cités pour leur installation et leur configuration de base. Seuls les éléments de configuration spécifiques à AFS sont mentionnés dans la suite de ce manuel.

2.2. Prérequis matériel

Le nombre de serveurs nécessaires pour une volumétrie et un trafic donnés dépendra :

- du type de données et de leurs tailles : documents, base de données, email, ...
- de la fréquence (mensuelle, hebdomadaire, quotidienne) et du type d'indexation (totale ou différentielle) en prenant en compte le nombre d'éléments qui ont changés entre deux indexations
- des besoins en haute disponibilité (redondance).

D'une façon générale, bien que tout type de matériel puisse être utilisé, du petit serveur mono-processeur d'entrée de gamme à des machines massivement multi-processeurs, AFS

tourne majoritairement sur des serveurs de type bi-processeurs x86 (avec 4Go de mémoire et un ou deux disques SCSI). Ce profil de serveur offre un rapport prix/performance inégalé, ainqi qu'une grande flexibilité de configuration par ajout de ressource.

2.2.1. Profils de serveurs

Afin d'aider le lecteur à définir le type de serveurs à mettre en oeuvre et leurs configurations respectives, pour chaque type de module nous indiquons les caractéristiques principales. Il est évident qu'il n'est pas nécessaire de mettre en place un serveur distinct par module et que tous les modules peuvent être rassemblés sur un seul et même serveur (ce qui sera le cas la plupart du temps), mais cette description des besoins en ressources matérielles de chaque composant permettra au lecteur de mieux cerner les besoins pour obtenir des performances optimales.

Pour le back-end :

- *URLs Manager* : il s'agit principalement d'un serveur de base de données et nécessite donc plutôt des disques rapides (SCSI 10.000 ou 15.000 RPM[5]) et de la mémoire ;
- *Document Manager* : son travail est de stocker les contenus crawlés à indexer. Il nécessite donc surtout des disques rapides (performance en I/O). Plusieurs solutions sont envisageables : NAS avec un serveur associé, petits serveurs avec disques SCSI rapides (configuration RAID). A titre d'exemple, le data center Antidot qui opère la solution AFS en ASP utilise des serveurs 2U avec des CPU de type PIII 1Ghz, 1Go de mémoire et 4 disques SCSI 146Go 10 000RPM en RAID5). Compte tenu de la performance des disques SATA actuels, des serveurs utilisant des disques performants de ce type sont tout à fait adaptés.
- *Index Manager* : nécessite une configuration équilibrée : disques et processeurs rapides.
- *Scheduler*, *Crawlers*, *Indexers*, *Word Manager* et *Language Checker* : utilisent surtout de la CPU. Des serveurs de type blade ou serveur 1U d'entrée de gamme avec un disque SATA et processeur rapide (dual core par exemple) conviennent parfaitement.

Pour le front -end :

- *RMS* : comme pour les *Crawlers* et les *Indexers* : de la puissance CPU.
- *Agents Répondeurs* : principalement de la CPU et de la mémoire, sachant que pour des index de taille importante (à partir de 500 000 documents) un disque rapide de type SCSI peut être nécessaire.
- *Snippet Server* : même profil que le *Document Manager* (du disque rapide).
- *Log Manager* et *Back-Office* : comme pour l'*URLs Manager*, il s'agit surtout de base de données.

2.2.2. Indications de volumétrie

Ces chiffres sont donnés à titre indicatifs et peuvent varier selon la taille de chacun des éléments de contenu indexés.

Pour 10 000 pages web : compter 2 Go d'espace de travail back-end pour stocker les pages crawlées et générer les index ; et compter 100 Mo au niveau front-end pourt stocker les index générés et les snippets.

En ce qui concerne les logs, compter 1Go par tranche de 5 millions de requêtes***.***

Un serveur bi-processeur de dernière génération permet d'indexer entre 2 et 4 millions de documents par jour (selon la complexité des traitements appliqués).

5 RPM (Round Per Minute) = vitesse de rotation exprimée en tours par minute