IBM WebSphere Application Server Version 5.1

Architecture, Sécurité et Répartition de charge

Présenté par:

Alexandre BOLE & Olivier MICHEL

Professeur responsable : M. Duris

Année 2003 – 2004

Table des matières

Introduction

Au coeur de l'actualité, les serveurs d'applications J2EE sont en concurrence avec la nouvelle plate-forme .NET de Microsoft. L'intérêt de J2EE est de rendre interopérable les composants développés dans le cadre des architectures distribuées, aussi appelées architectures multi-niveaux.

Dans le cadre du cours des Nouvelles Technologies Réseau, nous avons choisi de présenter le serveur d'application d'IBM WebSphere Application Server version 5.1. Le but est de vous fournir les informations nécessaires à la compréhension des différents modules. Étant composé d'un nombre importants de modules, nous avons choisi de nous concentrer sur la sécurité et la répartition de charge.

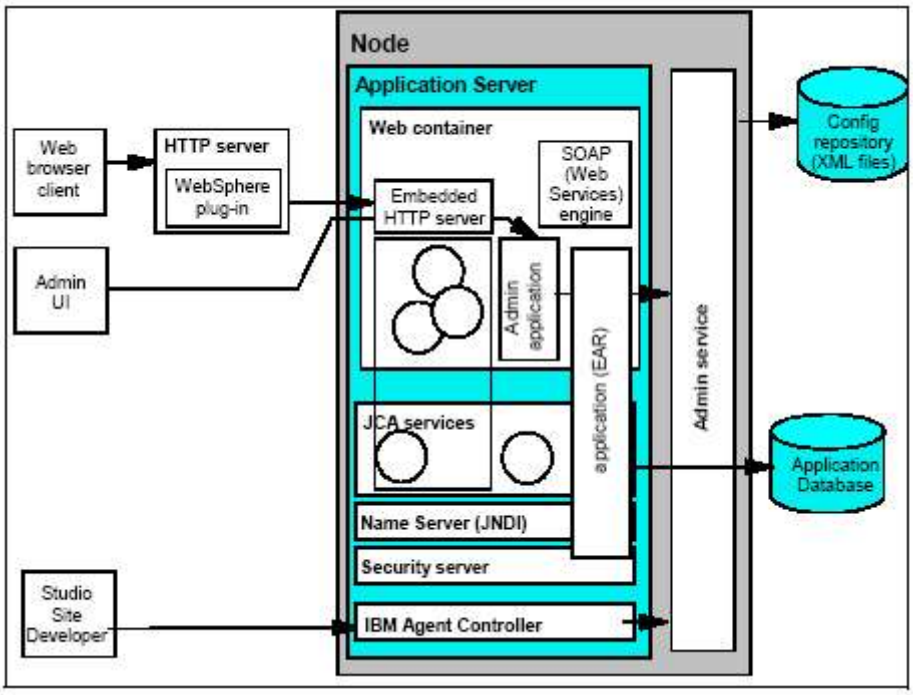
Dans un premier temps, nous présenterons l'architecture générale du serveur, puis les différentes configurations de sécurité disponible et enfin, la répartition de charge aux niveaux des serveurs Web, des containers JSP et EJB.

L'architecture du serveur WebSphere 5

La solution WebSphere se compose de 4 versions Express, Base, Entreprise et Network Deployment. Ces versions ont des architectures différentes et vont dépendre du besoin du client. Dans ce chapitre, nous allons décrire les 4 versions avec leurs spécificités et présenter les différents modules.

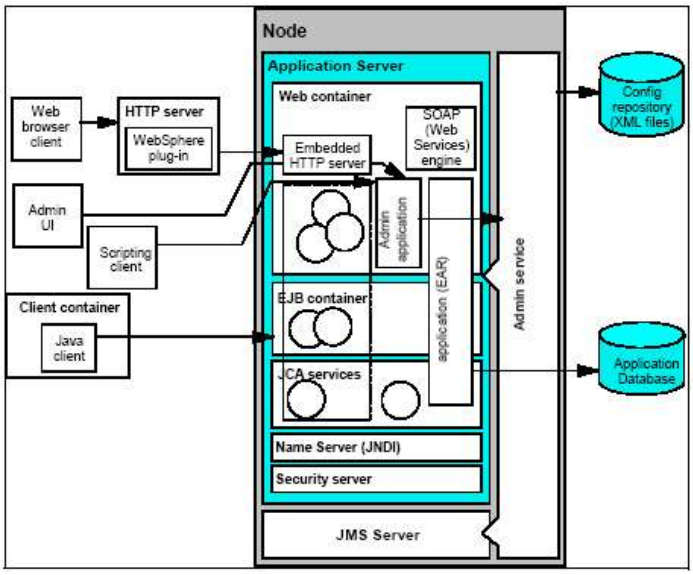
1. WebSphere Express

La version Express comme la Base ont des caractéristiques similaires puisqu'ils ne supportent que des environnements single-server. Ils ne permettent pas de réaliser une administration centralisée, ni de gestion de flux comme le load balancing. Par conséquent, chaque serveur d'application de type Express sera une entité unique contrairement aux versions Entreprise et Network. La version Express pourrait s'apparenter à un serveur d'application de type TOMCAT. Comme vous pourrez le constater sur le figure ci-dessous, cette version ne dispose pas de container EJB, ni de support pour le messaging JMS (Java Message Service). La différence entre un serveur tomcat et cette version est que cette architecture dispose d'un connecteur JCA. Il permet de s'interconnecter avec les systèmes d'information de l'entreprise de type SAP, SIEBEL ...



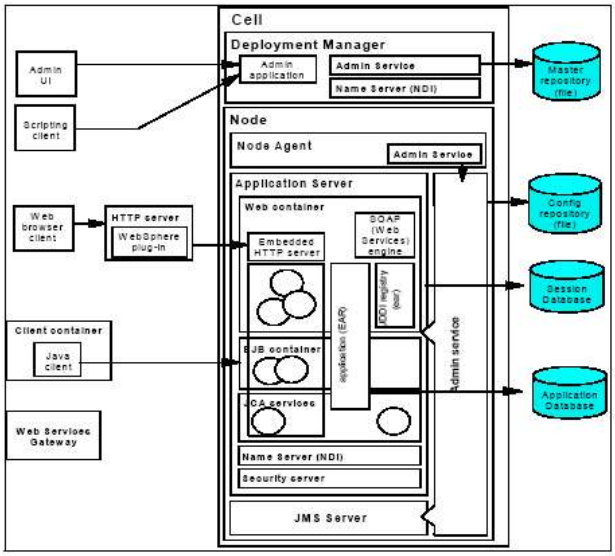
2. Websphere Base

La version Base est un serveur d'application J2EE, puisqu'il dispose d'un container EJB, du support du messaging. Malheureusement, cette version ne permet pas d'administration centralisée des serveurs, ni de gestion des flux (répartition de charges).



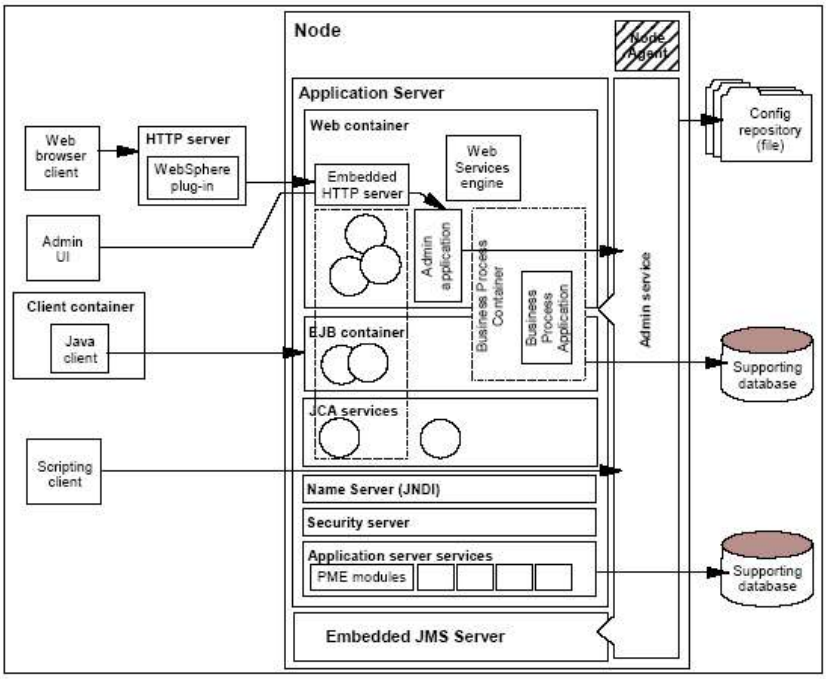
3. WebSphere Network Deployment

Cette version permet d'avoir une administration centralisée et une répartition de charges par la mise en place de load balancing, proxy cache et cluster. L'environnement Network Deployment est constitué de un ou plusieurs versions Base et d'un manager de déploiement. La version de base est ajoutée à ce qu'on appel une cellule qui est gérer par le Deployement Manager. Cette version comprend des fonctionnalités supplémentaires, comme la prise en charge des Web Services Private UDDI Registry et Web Services Gateway.



4. WebSphere Entreprise

Comme la version Network Deployement, l'Entreprise contient la version Base et le manager de déploiement. Elle comporte en plus le Programming Model Extensions (PME) pour le serveur d'application. Ces extensions sont composées de services supplémentaires, d'APIs, de Wizards de développement, et d'extensions de déploiement. Pour supporter ces différentes extensions, la version Entreprise dispose d'un Business Process Container sur le serveur d'application, ainsi qu'une console d'administration.



Les Cellules (Cells), Noeuds (Nodes) et serveurs (servers)

L'architecture des serveurs d'applications WebSphere est organisée sur les concepts de cellules, noeuds et serveurs.

1. Les serveurs (servers)

Sur une application Websphere, on distingue deux types de serveurs:

Les serveurs d'application : ont pour rôle d'exécuter les applications déployées par l'entreprise. Chaque serveur WebSphere se compose de un ou plusieurs serveurs d'application (processus). Dans la version express et base, chaque serveur d'application est indépendant.

Les serveurs JMS : Les versions Base, Network et Entreprise fournissent des serveurs JMS afin de pouvoir gérer le messaging. Pour la version base, le serveur JMS est intégré dans le serveur d'application, contrairement aux versions Network et Entreprise qui intègrent ce module sur une JVM séparée afin de garantir de meilleurs performances, ainsi qu'une résistance au crash.

2. Les noeuds (Nodes)

Un noeud est un groupe logique de processus serveurs qui partagent une configuration commune. Dans la majorité des cas, un noeud est associé avec une installation physique du serveur d'application WebSphere. Dans les versions Base et Express, il ne peut y avoir qu'un seul node. Pour les autres versions, l'intérêt du noeud est de permettre une administration centralisée, ainsi qu'une répartition de charge entre les noeuds. Cette configuration est obtenue grâce au node agent qui réalise la communication entre les serveurs d'applications à l'intérieur d'un noeud.

3. Les cellules (Cells)

Une cellule est un groupe de noeuds qui permet d'obtenir un domaine d'administration (comme pour les réseau NT par exemple). Dans les versions base et express, une cellule ne peut contenir qu'un seul noeud (une seule installation de WebSphere). Avec les versions Network et Entreprise, une cellule peut comporter plusieurs noeuds avec une administration centralisée. Pour permettre cette centralisation de l'administration, la configuration est stockée dans un repository. Ce repository est géré par le processus Deployement Manager intégré dans chaque serveur d'application WebSphere qui réalisent une copie du repository en locale.

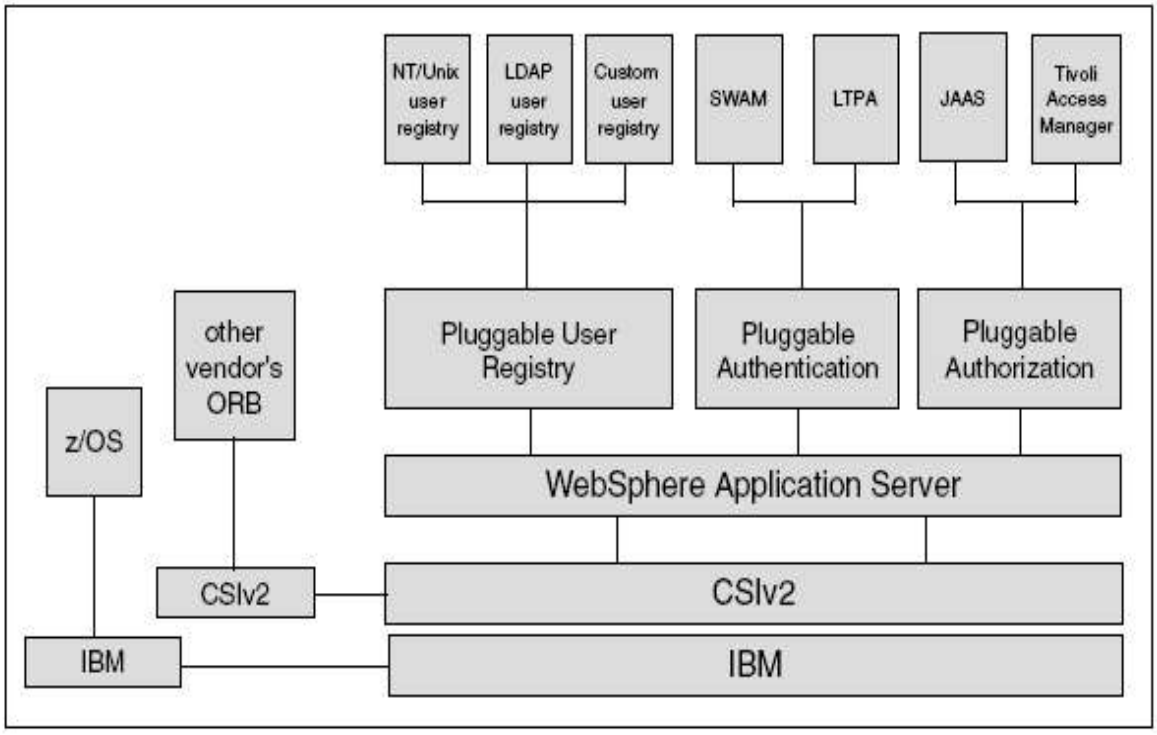
La sécurité sous WebSphere

Nous avons jusqu'à maintenant présenté l'architecture générale de WebSphere. Nous allons vou présenter une partie importante sur la sécurité. En effet l'architecture de sécurité est vraiment importante dans le cadre d'un serveur d'application qui sert d'interface avec l'extérieur qui est par défaut non sécurisé. Le serveur d'application IBM WebSphere V5 respecte entièrement la norme J2EE 1.3. Il implémente les services de sécurité requit par cette norme. Les composants de sécurité forment la partie essentielle de l'architecture de WebSphere, nous allons maintenant présenter d'une manière générale ces composants.

1. Présentation générale de la sécurité sous WebSphere

Comme nous le disions, WebSphere suit la norme J2EE 1.3, ce qui sous entend qu'il utilise un modèle de sécurité déclaratif qui peut être suivis par n'importe quelle application extérieur. Cette norme va permettre aux autres applications de venir se « plugger » au serveur d'application étendant ainsi ses fonctionnalités. Cette normalisation des interfaces permet à chaque implémentation du serveur WebSphere d'être différentes au niveau des choix de sécurité.

De base le serveur WebSphere fournit un certain nombre de composant de sécurité. L'architecture globale est la suivante :



La sécurité sous WebSphere peut être découpées en deux catégories de sécurité : la sécurité globale et applicative.

La **sécurité globale** spécifie la configuration de la sécurité globale pour un domaine de serveur. Ce type de sécurité s'appliquera a toutes les applications fonctionnant sur le serveur. Elle définira le type de registre utilisateur utilisé, les méthodes d'authentification et ainsi de suite. Toutes ces configurations sont accessibles depuis la console d'administration de WebSphere. On pourra noter que toutes les configurations données dans cette console agiront comme valeur par défaut pour toutes les applications.

La **sécurité applicative** définit les besoins spécifiques d'une application du serveur. Généralement ce type de sécurité permet de spécialiser ou complémenter des configurations générales et peut aussi permettre de contourner certaines configurations. Typiquement un exemple de ce type de configuration est que si nous avons définis dans la configuration globale que nous nous authentifions grâce au protocole SWAM par défaut, il est possible pour une application de redéfinir la méthode d'authentification à LTPA. De même, si l'application gère des ressources spécifiques, différents types d'utilisateur,... elle peut redéfinir des droits qui lui seront propre. La sécurité applicative est administrée lors du développement grâce à l'utilitaire *AAT* pour Application Assembly Tool et en cours de fonctionnement en utilisant la console d'administration ainsi que l'utilitaire *wsadmin.* Il faut tout de même noter qu'une application ne peut utiliser que des mécanismes de sécurité « pluggable » dans l'architecture WebSphere (suivant la norme J2EE).

Nous avons jusqu'à présent décrit la sécurité globale de WebSphere, et nous nous sommes attachés à décrire les types de sécurité prévus pour l'architecture WebSphere. Nous allons maintenant présenter les composants de sécurité proposés de base par WebSphere.

2. Les registres utilisateurs (Pluggable User Registry)

Les registres utilisateur stockent les noms d'utilisateurs et le nom des groupes d'utilisateurs qui sont utilisés principalement lors de des mécanismes d'authentification et d'autorisation sur le serveur WebSphere. Dans ces registres sont inscrit des données propres aux utilisateurs. Ces données sont dans un premier temps descriptifs et autorisant, car ils permettent de décrire le profils de l'utilisateur définissant sont niveau d'importance pouvant être utilisé dans les applications. On note trois types de registres de base sur le serveur WebSphere: *le LDAP user registry, Local operating system user registry,* et *Custom user registry*

WebSphere peut donc utiliser ces trois types de registres. Toutefois, WebSphere ne peut utiliser plusieurs registres utilisateurs en même temps pour l'authentification des utilisateurs.

2.1. Local operating system user registry

Cette configuration de registre permet d'utiliser le système d'exploitation sur lequel est exécuté WebSphere. Concrètement, WebSphere va interroger le système d'exploitation : Linux Windows (NT ou 2000) pour extraire les noms et groupes d'utilisateurs. Il faut bien être informé que même si WebSphere utilise les noms contenus dans les registres systèmes, il n'en utilise pas entièrement la hiérarchie de droit et il se peut qu'un utilisateur par le biais de WebSphere acquiert plus de droit que normalement.

2.2. LDAP user registry

Dans la plupart des utilisations de WebSphere, le registre d'utilisateurs LDAP est recommandé. En effet un serveur d'application est a priori utilisé afin de brasser un nombre de client. Il n'est pas logiquement concevable d'utiliser un registre d'utilisateur local dans ces cas.. De plus la plupart des serveur LDAP disponibles sur le marché sont maintenant suffisamment équipé de mécanisme de sécurité pour être fiable et être utilisable sans a priori avec WebSphere. Cependant WebSphere ne supporte pas tous les serveurs LDAP du marché. En effet celui ci offre le support de quelques serveurs : LDAP servers: IBM SecureWay Directory ,Netscape LDAP Server, Lotus Domino LDAP Server, Microsoft Active Directory. WebSphere fournit aussi la possibilité d'utiliser d'autres serveurs LDAP. En effet, WebSphere fournit aux administrateurs une flexibilité sur les paramètres de recherche de serveurs assez large pour prendre en compte un maximum de serveurs.

2.3. Custom user registry

Ce module laisse une porte ouverte vers une implémentation personnalisée d'un registre utilisateur. L'API de WebSphere définie une interface JAVA (*UserRegistry*) qui doit être utilisée pour écrire le registre personnalisé. Cette interface permet aussi de définir tous les accès virtuels aux données sauvegardées que se soit dans un fichier sur le serveur ou depuis un serveur de base de données. Ce module va dans un sens permettre l'interaction entre le serveur d'application et les modules d'authentification par exemple et notre implémentation normée du registre.

3. Les modules d'authentifications (Pluggable Authentification)

L'authentification est une série de règles de sécurité qui permettent d'identifier le client avec qui ont discute. Cette authentification permet simplement d'établir un état persistant indisponible sur des protocoles réseaux comme HTTP par exemple. Sur WebSphere, la phase d'authentification va permettre de mettre au point un certain nombre de règles qui vont par la suite être réutilisées lors des traitements dans l'application JAVA.

L'authentification collabore en duo avec le registre utilisateur de WebSphere. En effet afin d'identifier le client qui va se connecter sur le système, il faudra que le mécanisme d'authentification puisse parcourir le registre d'utilisateurs, lors d'une demande d'authentification. Il faut aussi que le mécanisme créé un « credential[1] » qui identifie les droits

1 littéralement la traduction de « credential » est « Justificatif d'identité » ou « référence ». Nous utiliserons dans ce rapport le mot « credential » afin de bien marquer la notion de droit sur les clients, serveurs, EBJ, ...

associés à un utilisateur ou a un groupe d'utilisateurs. Tous les « credentials » ne sont pas égaux. En fait, ils dépendent de beaucoup de paramètre inhérent au protocole d'authentification. WebSphere considère que dès qu'un « credential » est créé, alors le client est authentifié. Il reste que certaines fonctionnalités dépendantes du protocole d'authentification ne pourront pas être utilisées par l'application. WebSphere ne supporte que la configuration d'un protocole d'authentification à la fois et est configuré avec la définition de la *sécurité globale*.

WebSphere fournit deux protocoles d'authentification de base : Simple WebSphere Athentification Mechanism (SWAM) et Lightweight Third Party Authentication (LTPA). Ces deux protocoles diffèrent essentiellement sur leur comportement avec des applications réparties. Nous allons maintenant présenter ces deux protocoles.

3.1.SWAM (Simple WebSphere Authentication Mechanism)

SWAM pour Simple WebSphere Authentification Mechanism est le protocole d'authentification qui s'adresse aux applications solitaires et simples. Solitaire et simple sont des qualificatifs utilisées ici pour décrire une application qui sera lancée dans un contexte simple (a but unique) et dans un environnement non répartie. En effet SWAM inclus une restriction sur la transmission des « credentials ». Dans d'autres termes cela signifie que si une servlet ou un EJB dans une application serveur 1 appelle une méthode distante sur un EJB ou une servlet fonctionnant dans le contexte d'une deuxième application serveur, alors l'identité de l'appelant (dans la première application serveur) ne sera pas transmis a la deuxième application. Ce qui est réellement transmis est un « unauthenticated credential » pour identifier le processus appelant et de ce fait, il peut engendrer des échecs d'authentification lors de la tentative de communication avec l'EJB distant par exemple. Ces échecs dépendrons de la politique de sécurité des EJB, servlet,....

De plus depuis que SWAM a été définie comme protocole pour les applications « solitaires », les méthodes SSO ne sont plus supportées. SSO signifie « Single Sign ON ». Le mécanisme SSO permet a l'utilisateur de ne s'identifier qu'une et une unique fois auprès du serveur. Avec un principe de session, l'application serveur peut effectuer des transactions sans redemander une authentification.

Un exemple d'utilisation du SSO se fait dans le milieu bancaire. En effet, imaginez que vous entriez sur le site de votre banque et qu'à chaque requête (consultation de vos comptes, mise a jours, ...) vous deviez entrer encore et encore votre nom d'utilisateur et votre mot de passe. La navigation deviendrait très vite impossible, la sécurité difficile a mettre en place (si l'utilisateur se trompe de mot de passe ...). SSO normalise la notion de session utilisateur avec seulement une authentification.

En conclusion à ce protocole, l'absence du support de SSO et de la transmission des droits en fait un protocole adapté dans le cadre d'application serveur très simple et ne nécessitant quasiment aucune notion de droits, ou environnement partagé. De plus à l'authentification, par défaut, SWAM se base sur un mécanisme d'ID de session pour authentification de l'utilisateur ce qui n'est pas vraiment sécurisé, c'est pourquoi il est conseillé d'utiliser SSL avec SWAM.

3.2.LTPA (Light Weight Third Party Authentication)

LTPA, pour Lightweight Third Party Authentication, est tout l'inverse du protocole SWAM. En effet celui ci est entierement destiné a une utilisation dans des applications multiples, et réparties. Il supporte la transmission des « credential » ainsi que SSO. LTPA est capable de supporter la sécurité des applications réparties a travers l'utilisation de la cryptographie. Ce qui permet a LTPA de crypter, signer et transmettre de manière sécurisé l'ensemble des données d'authentification. Bien entendu l'opération de décryptage et vérification des données est aussi possible.

LTPA requiert simplement l'utilisation d'un registre d'utilisateur partagé et centralisé de type LDAP.

Le tableau suivant est un récapitulatif des possibilités des deux protocoles d'authentification fournit par WebSphere

Protocole	Forwardable user credentials	SSO	Local OS user registry	LDAP user registry	Custom user registry
SWAM	Non	Non	Oui	Oui	Oui
LTPA	Oui	Oui	Oui	Oui	non

Tableau : Résumé des protocoles

Les versions futures de WebSphere supporterons l'authentification par Kerberos afin de fournir un plus large choix ainsi qu'un mécanisme de norme d'industrie pour l'authentification.

4. Les modules d'autorisation (Pluggable Authorization)

Les modules d'autorisation standard de WebSphere sont basés sur la spécification de la J2EE. Ces modules sont aussi basée sur les services proposés par une extension de l'architecture de sécurité de la plate forme JAVA appelée JAAS (Java Authentification and Authorization Services). Cette extension permet l'ajout d'une authentification et d'une gestion des droits par ACL des utilisateurs. Un autre mécanisme d'autorisation est disponible. Il s'agit de Tivoli Access Manager qui au même titre que JAAS permet d'authentifier et d'autoriser les utilisateurs à interagir avec le système, les applications, ...

Nous allons maintenant détailler ces deux procédés qui sont deux des éléments clés de la sécurité de WebSphere.

4.1.JAAS : Java Authentification and Authorization Services

En théorie, JAAS est une série de paquetages qui implémente un service d'authentification et d'autorisation des utilisateurs. JAAS implémente une version standard du framework PAM (Pluggable Authentification Module) et supporte une autorisation par utilisateurs. Dans la pratique, JAAS est le standard de sécurité de JAVA (depuis qu'il a été intégré à la JDK 1.4).

D'un point de vue architecturale, JAAS implémente une version de PAM. Ce framework est un framework à l'architecture modulaire qui est spécialisé dans les échanges sans état entre composants. Il permet aussi l'utilisation et l'ajout de multiple technologies d'authentification / autorisation sans changement ou interférence avec les protocoles d'authentifications précédemment mis en place. Ainsi PAM peut être utilisé comme service d'identification avec de nombreuses technologies d'authentifications : RSA, DCE, Kerberos, S/Key, et aussi les systèmes d'authentification basés sur les « smart card » java.

JAAS est découpé en une architecture de type « plugin », utilisant des modules de codes qui implémentent une série d'interfaces. Ces modules permettent aux applications JAVA pour rester indépendant du protocole d'authentification utilisé. Ainsi l'ajout, la suppression ou la modification du protocole d'authentification peuvent être effectué sans modification de l'application ou recompilation du code.

L'API de JAAS est très complète et nous allons maintenant détailler les interfaces principales de JAAS :

- **Callback** – Les implémentations de cette interface permettent d'encapsuler les informations (nom d'utilisateurs, mots de passe, erreur et messages d'avertissement) qui sont échangées entre les services de sécurités et un *CallbackHandler* (expliqué juste après).
- **CallbackHandler** – Permet de faciliter la communication entre une application et les service de sécurités. Le CallbackHandler est spécifique à l'application.
- **LoginContext** – Le LoginContext fournit les méthodes basiques utilisées afin d'authentifier d'une manière neutre un utilisateur, une autre application, ...
- **LoginModule** – Les protocoles d'authentification fournissent une implémentation de cette interface afin de fournir un type particulier d'authentification a travers un module ajouté.
- **Principal** – Représente une entité unique : utilisateur, groupe d'utilisateur, organisation, mot de passe, numéros de sécurité sociale, ... qui peut être authentifié.
- **Subject** – Représente un groupement d'informations pour une entité. Une ou plusieurs entités sont liées à un sujet. Chaque entité devient une identité du sujet. Chaque sujet maintient aussi des informations sur la sécurité : (passwords et clé de cryptage, par exemple).

4.2.Tivoli Access Manager

TIVOLI Access Manager est une solution complète d'authentification et d'autorisation pour permettre le déploiement rapide d'applications Web sécurisées. Elle gère le contrôle d'accès de tout utilisateur et navigateur Internet aux serveurs et applications web.

Ce logiciel est conçu pour :

- Fournir des règles de contrôle d'accès faciles à gérer.
- Proposer un système d'authentification et d'autorisation extensible.
- Permettre à la fois un accès distant personnalisé et sécurisé.
- Garantir une authentification unique (Single Sign On) pour l'accès à des ressources Web multiples.
- Supporter les infrastructures à clés publiques..

5. Les autres composant de la sécurité sous WebSphere

Nous allons maintenant présenter une série de composants utilisés pour la sécurité interne au serveur d'application WebSphere V5.

5.1. Security Server

Security server est un composant du serveur d'application WebSphere qui est exécuté dans chaque processus serveur. Si plusieurs applications serveurs sont exécutées sur un unique serveur, alors il existe une instance par processus. Ce composant est responsable de l'authentification et du maintient des sessions utilisateurs et il collabore avec le processus d'autorisation ainsi que le gestionnaire de registres utilisateurs.

5.2. Security collaborators

Les « Security collaborator » sont des processus serveurs responsables de l'exécution des contraintes de sécurité spécifiées dans les configurations générales du serveur. Ils communiquent avec le « security server » à chaque demande d'authentification ou demande d'autorisation. On peut distinguer deux types de collaborateurs :

Web security collaborator inclus dans le module Web.
Les « Web collaborator » fournissent les services suivant:
- Vérifie l'authentification
- Effectue l'autorisation en accord avec les configurations spécifiées dans les descripteur de déploiement
- Effectue des logs de sécurité.

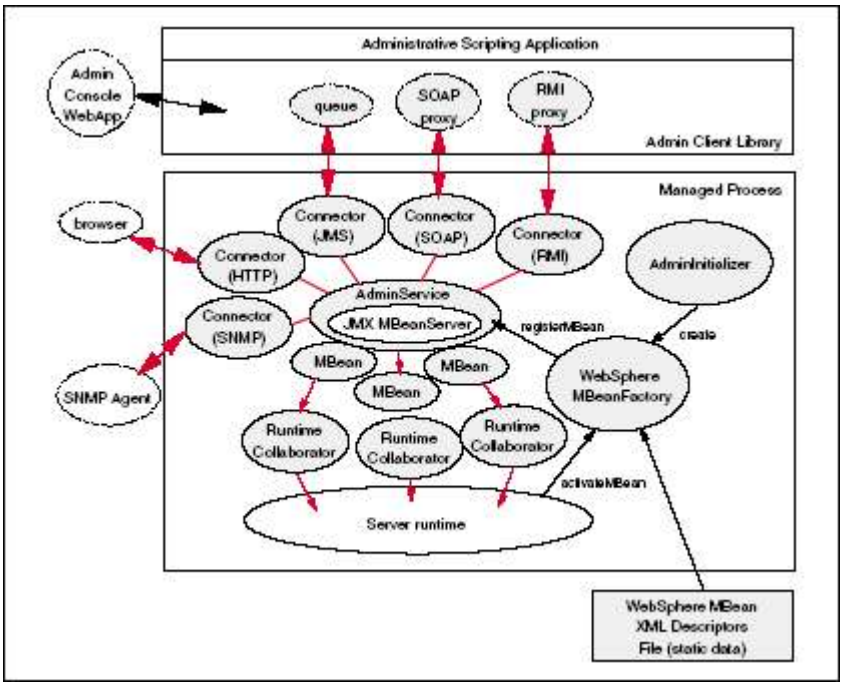
EJB security collaborator inclus dans le conteneur EJB.
Les collaborateurs EJB utilise les protocoles CSIv2 et SAS pour authentifier les requêtes des clients JAVA auprès des EJB. Ces collaborateurs fournissent aussi les fonctionnalités suivantes :
- Vérifie l'authentification des clients
- Supporte la communication avec le registre utilisateur
- Effectue des logs de sécurité.
- Communique avec l'ORB a l'aide du protocole CSIv2

5.3. JMX MBeans

JMX, pour Java Management Extension, est une série de nouvelles interfaces et de java Beans qui permettent de gérer la configuration et la mise a jour des applications fonctionnant sous WebSphere depuis une application indépendante du système normal d'administration du serveur d'application (console administrative ou le programme wsadmin sous websphere).

JMX utilise le schéma d'implémentation de Tivoli et fournit une interface cliente ou un utilitaire de script permettant de gérer WebSphere en entier à travers l'utilisation des « Mbeans » (Management Beans) aussi fournit par JMX. Généralement JMX est utilisé afin de géré les différentes taches, processus ou application de WebSphere.

Voici un schéma présentant les interactions globales entre le noyau du service d'adminstration de WebSphere utilisant JMX et les autres composants :



La répartition de charge avec des serveurs d'applications WebSphere

1. La répartition de charge entre les serveurs Web - Network dispatcher

L'un des points fort de WebSphere est de permettre une répartition de charge entre les différents serveurs HTTP. Cette gestion de la charge est réalisée par un composant appelé *Network Dispatcher.*

Le composant de Network Dispatcher permet de gérer le flux du trafic réseau, réduisant la surcharge via des goulots et équilibrant la charge sur divers autres services. Ces serveurs peuvent être soit des points d'entrée sur Internet, soit des serveurs de réseau interne.

Pour effectuer cette répartition, le composant Network Dispatcher intercepte les demandes de données provenant des clients et les transmet au serveur le mieux à même d'y répondre. En d'autres termes, il équilibre la charge des demandes entrantes sur un ensemble défini de machines qui traitent le même type de demandes. Le composant Network Dispatcher peut répartir les demandes entre différents types de serveurs, y compris les serveurs HTTP et les machines Caching Proxy. Ce composant permet de définir des règles spécifiant les critères utilisés pour déterminer le serveur le mieux approprié pour traiter une demande.

Le Network dispatcher se composent de plusieurs modules:

- Dispatcher
- Content Based Routing (CBR)
- Site Selector
- Cisco CSS Controlller
- Nortel Alteon Controller

Dans les paragraphes suivants, nous allons présenter les différents modules qui composent Network dispatcher.

1.1.Dispatcher

Le composant dispatcher permet de réaliser un rééquilibrage de charge sur un réseau local (LAN) ou un réseau étendu (WAN) pour les protocoles HTTP, FTP, SSL, NNTP, IMAP, POP, SMTP et Telnet. Le dispatcher réalise un forward vers les serveurs suivant plusieurs méthodes MAC, NAT (Network Address Translation) ou NAPT (Network Adress Port Translation) et CBR.

- **Méthode d'acheminement MAC** : Cette méthode d'acheminement permet à Dispatcher d'équilibrer la charge de la demande entrante adressée au serveur. Le serveur HTTP renvoie la réponse directement au client sans l'intervention de Dispatcher.

- **Méthode d'acheminement NAT/NAPT** : L'utilisation de la fonction NAT (Network Address Translation)/ NAPT (conversion des ports d'adresse réseau) élimine la nécessité pour les serveurs principaux de se trouver sur un

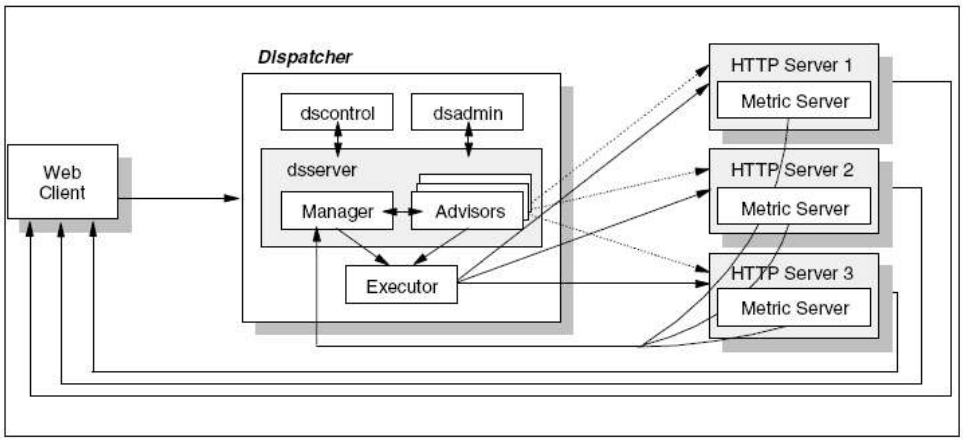
réseau local. Cette méthode d'acheminement permet à Dispatcher d'équilibrer la charge de la demande entrante adressée au serveur. Le serveur renvoie la réponse à Dispatcher. La machine Dispatcher renvoie ensuite la réponse au client.

- **Méthode d'acheminement Content-Based Routing** : Sans Caching Proxy[2], le composant Dispatcher permet d'exécuter la fonction CBR (content-based routing) pour HTTP (avec la règle de type de contenu) et HTTPS (avec l'affinité des ID de session SSL). Pour le trafic HTTP et HTTPS, le composant Dispatcher peut fournir une fonction CBR (content-based routing) plus rapide que le composant CBR. Cette méthode d'acheminement permet à Dispatcher d'équilibrer la charge de la demande entrante adressée au serveur. Le serveur renvoie la réponse à Dispatcher. La machine Dispatcher renvoie ensuite la réponse au client.

Le dispatcher se compose de plusieurs modules. Le coeur du système est géré par l'Executor. Il s'agit du noyau qui a pour fonction d'examiner l'entête de chaque paquet et de savoir si celui-ci représente une connexion existante ou une nouvelle connexion. Pour gérer cette fonctionnalité, l'Executor dispose d'une table de connexion en mémoire permettant de déterminer l'adresse du serveur sur lequel les paquets doivent être transmis si la connexion existe.

Pour effectuer le routage de paquet, l'executor se base sur les informations fournis par le Manager. Celui-ci détermine la charge des serveurs grâce au module Advisor. Advisor est un client léger qui exécute des commandes sur les serveurs afin de déterminer le statut de ceux-ci. Si la requête envoyée est effectuée avec succès, l'advisor considère que le serveur est Up. Le problème de cette technique est quelle ne permet pas de déterminer la charge réel du système, mais seulement de savoir si le système répond encore.

Pour combler cette lacune, un serveur léger appelé Metric Server peut être déployé sur chaque serveur devant bénéficier de la répartition de charge. Ces serveurs transmettent régulièrement au manager du Dispatcher le taux de charge de la machine sur laquelle il se trouve (voir figure ci-dessous). Cette méthode permet d'avoir un suivi plus précis des serveurs et une répartition équitable en fonction du taux d'occupation.



Pour administrer le dispatcher, vous disposez de deux techniques, l'une en ligne de commande grâce au module dscontrol et l'autre avec une interface graphique grâce au module dsadmin.

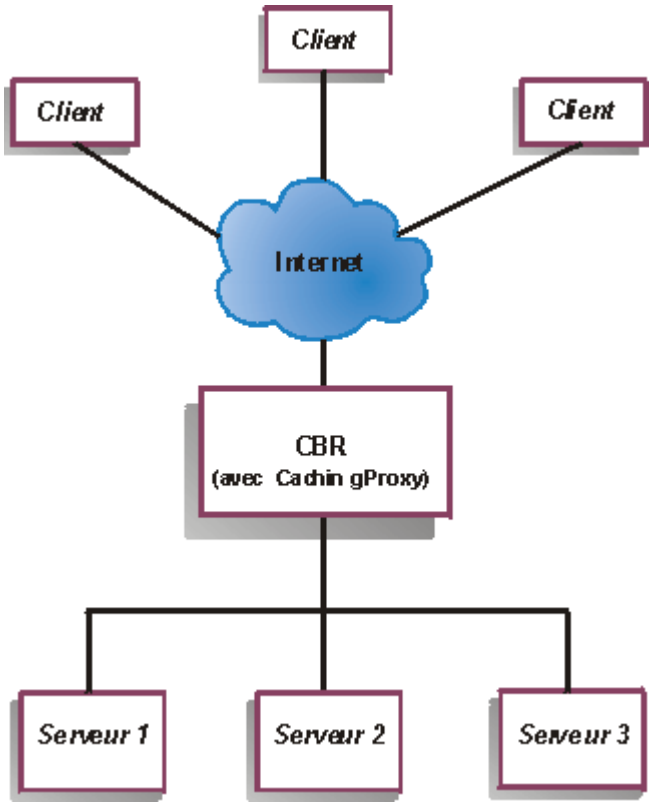
2 Serveur proxy fournit par WebSphere. Celui-ci fait l'objet d'un paragraphe quelques pages plus loin

1.2.Content Based Routing (CBR)

CBR coopère avec Caching Proxy pour relayer les demandes des clients aux serveurs HTTP ou HTTPS (SSL) indiqués. Il permet de manipuler les détails de la mémoire cache pour accélérer le rappel des documents Web. CBR permet de spécifier un groupe de serveurs qui doit prendre en charge une demande en fonction de son contenu. CBR permet également d'indiquer plusieurs serveurs pour chaque type de demande. Par conséquent, la charge des demandes peut être répartie pour obtenir une réponse optimale du client. CBR peut aussi détecter les incidents qui se produisent sur un serveur et arrêter d'acheminer des demandes vers ce dernier. L'algorithme d'équilibrage de charge utilisé par le composant CBR est identique à l'algorithme utilisé par le composant Dispatcher.
Lorsqu'une demande est reçue par Caching Proxy, elle est comparée aux règles qui ont été définies dans le composant CBR. En cas de correspondance, l'un des serveurs associés à cette règle est désigné pour prendre en charge la demande. Caching Proxy continue alors son traitement normalement pour acheminer la demande vers le serveur sélectionné.

CBR offre les mêmes fonctions que Dispatcher à l'exception des fonctions de haute disponibilité, de réseau étendu et de quelques commandes de configuration.

Caching Proxy doit être en fonction pour permettre à CBR d'équilibrer la charge des demandes des clients.

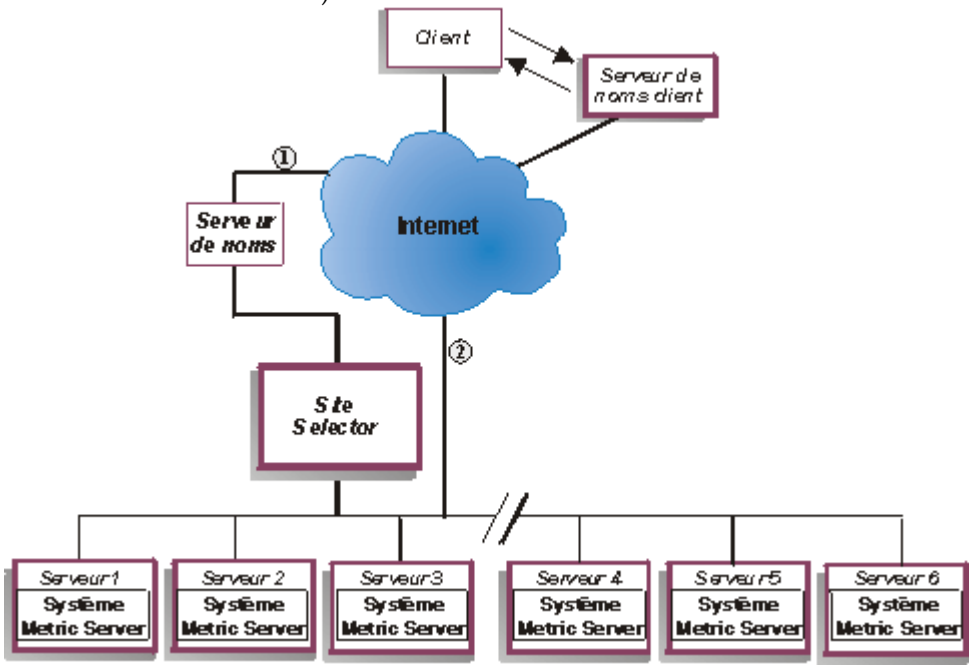


1.3.Site Selector

Site Selector fonctionne avec d'autres serveurs de noms pour équilibrer la charge sur un groupe de serveurs à l'aide des mesures et des pondérations recueillies. Il permet de créer une configuration de site pour assurer l'équilibrage de charge sur un groupe de serveurs sur la base du nom de domaine utilisé pour la demande d'un client.

Un client envoie une demande de résolution de nom de domaine à un serveur de noms appartenant au réseau. Le serveur de noms achemine la demande au poste Site Selector. Site Selector résout le nom de domaine en adresse IP de l'un des serveurs qui a été configuré sous le nom du site. Site Selector renvoie l'adresse IP du serveur sélectionné au serveur de noms. Le serveur de noms renvoie l'adresse IP au client.

Metric Server est un composant de Network Dispatcher décrit précédemment qui surveille le système et doit être installé sur chaque serveur dont la charge doit être équilibrée. Metric Server permet à Site Selector de surveiller le niveau d'activité d'un serveur, de détecter le moment où un serveur est le moins chargé et de détecter un serveur défaillant. En personnalisant les fichiers scripts de mesure du système, vous pouvez choisir le type de mesure utilisé pour évaluer la charge. Site Selector peut être configuré en fonction de chaque environnement, en tenant compte de facteurs tels que la fréquence des accès, le nombre total d'utilisateurs et les différents types d'accès (requêtes à forte ou faible consommation de ressources - CPU).



La figure ci-dessus illustre un site utilisant le composant Site Selector pour répondre aux demandes. Serveur 1, Serveur 2 et Serveur 3 sont des serveurs locaux. Les serveurs 4, 5 et 6 sont des serveurs distants.

Un client envoie une demande de résolution de nom de domaine à un serveur de noms. Le serveur de noms achemine la demande au poste Site Selector (chemin d'accès 1) via DNS. Site Selector résout ensuite le nom de domaine en adresse IP de l'un des serveurs. Site Selector renvoie l'adresse IP du serveur sélectionné au serveur de noms. Le serveur de noms renvoie l'adresse IP au client.
Lorsque le client reçoit l'adresse IP du serveur, il achemine les demandes suivantes directement au serveur sélectionné (chemin d'accès 2).

1.4.Cisco CSS Controller and Nortel Alteon Controller

Cisco CSS Controller et Nortel Alteon Controller constituent une solution complémentaire. Ces deux solutions fournissent des fonctions d'acheminement de paquets et de routage de contenu permettant de déterminer la disponibilité des serveurs principaux, des applications et des bases de données.

La fonction de ces contrôleurs fait appel au gestionnaire de Network Dispatcher et à Metric Server pour déterminer la charge des serveurs principaux, des applications et des bases de données. Ces contrôleurs utilisent ces informations pour générer les mesures de pondération, qu'il envoie aux serveurs Cisco CSS Switch ou Nortel pour la sélection du serveur optimal, l'optimisation de la charge et la tolérance aux pannes. es serveurs Cisco CSS Switch ou Nortel prennent les décisions d'équilibrage de charge en fonction des critères définis par l'utilisateur.

Les contrôleurs suivent de nombreux critères, dont :

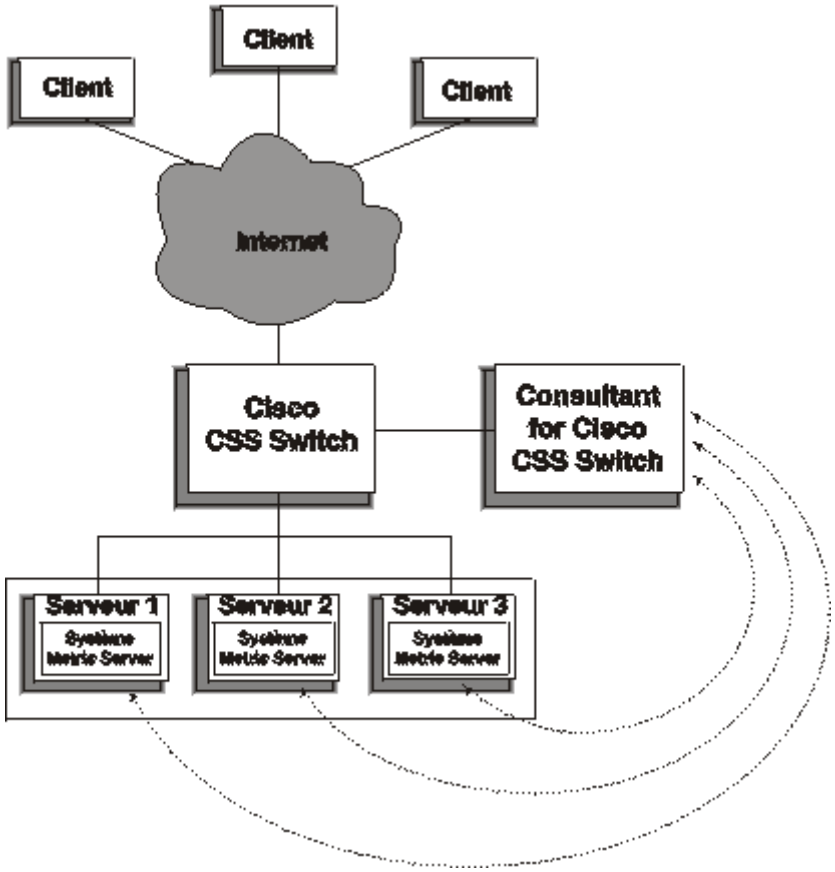
- les connexions actives et nouvelles,
- la disponibilité des applications et des bases de données,
- l'utilisation du CPU,
- l'utilisation de la mémoire,
- les mesures du serveur personnalisables par l'utilisateur.

Lorsqu'un serveur Cisco CSS Switch (ou Nortel), sans Cisco Controller, détermine le taux de charge d'un serveur de contenu, il utilise les temps de réponse aux demandes de contenu ou d'autres mesures de réseau. Avec Cisco Controller, le serveur Cisco CSS Switch se décharge de cette tâche sur le controleur Cisco Controller. Cisco Controller influence la pondération du serveur et active ou suspend un serveur, selon le cas, lorsque le serveur devient disponible ou indisponible.

Cisco Controller ou Nortel Controller:

- Utilise une interface SNMP pour obtenir des informations de connexion à partir du serveur Cisco CSS Switch ou Nortel Switch.
- Utilise les données d'entrée du conseiller pour analyser les informations de connexion.
- Utilise les informations Metric Server pour analyser la santé relative du serveur .
- Génère des pondérations pour chaque serveur de la configuration.

Les pondérations définies s'appliquent à tous les serveurs connectés sur un même port. Pour chaque port, les demandes sont réparties entre les serveurs selon la pondération. Par exemple, si un serveur a une pondération (paramètre Weight) de 10 et un autre de 5, le premier recevra deux fois plus de demandes que le second. Ces pondérations sont fournies au switch Cisco CSS Switch ou Nortel Switch avec SNMP. Lorsque la valeur de pondération d'un serveur est augmentée, le serveur Cisco CSS Switch ou Nortel Switch dirige davantage de demandes vers ce serveur.



2. Le serveur Caching Proxy

Les composants Caching Proxy fournissent le serveur proxy avec mémoire cache. Ce serveur proxy de base peut être configuré comme serveur proxy avec mémoire cache inverse ou comme serveur proxy avec mémoire cache d'acheminement, afin d'améliorer les temps de réponse.

Le composant Caching Proxy intercepte les demandes de données émanant d'un client, extrait les informations demandées sur les hôtes de données et les fournit au client. Le plus souvent, les demandes concernent des documents stockés sur des serveurs Web (également appelés *serveurs d'origine* ou *hôtes de données*) et fournis via le protocole HTTP (Hypertext Transfer Protocol). Vous pouvez toutefois configurer Caching Proxy de sorte qu'il accepte d'autres protocoles, tels que FTP (File Transfer Protocol) et Gopher.

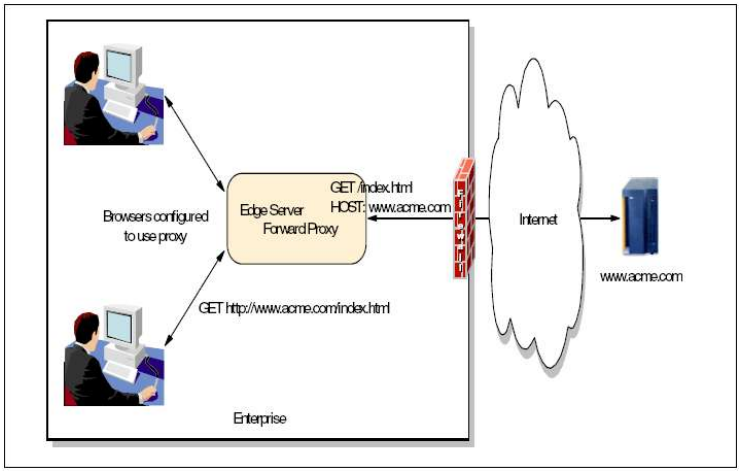
Caching Proxy stocke les données dans une mémoire cache locale avant de les fournir au demandeur. Les données pouvant être mises en mémoire cache incluent des pages Web statiques et des fichiers JSP comportant des parties générées dynamiquement. La mise en mémoire cache permet à Caching Proxy de satisfaire les futures demandes concernant les mêmes données, directement depuis la mémoire cache locale, ce qui nécessite bien moins de temps qu'une nouvelle connexion sur l'hôte de données.

Dans les parties suivantes, nous allons vous décrire les différents types de serveur proxy.

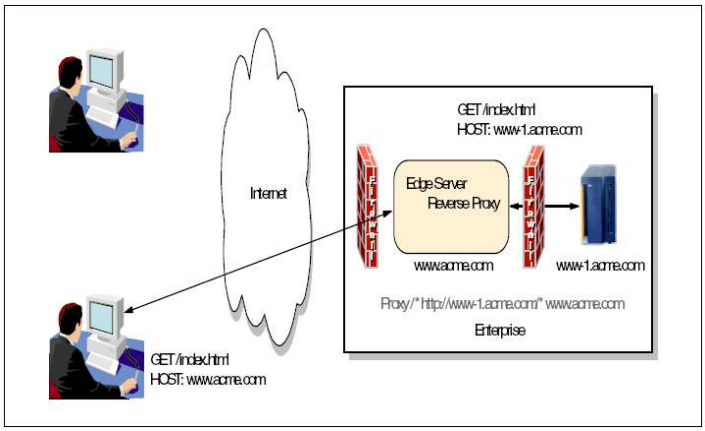
2.1.Forward proxy et Transparent Proxy

Le forward Proxy permet de fournir un accès à internet en limitant les requêtes vers l'extérieur sur les pages souvent consultées. Lorsque l'on utilise cette configuration, il est nécessaire de configurer chaque client afin de leur indiqué qu'ils doivent utiliser le proxy pour accéder à l'extérieur. Par conséquent, lorsqu'un client effectue une requête vers l'extérieur sur une page qui n'a jamais été demandée, le caching proxy demande au serveur distant de lui transmettre les informations demandées par client. Puis, il va stocker les données dans son cache afin d'optimiser les requêtes pour les prochains clients. Lorsque quelqu'un va demander à nouveau cette requête, le proxy Cache va s'apercevoir qu'il possède déjà en cache les informations sur cette page. Il transmet directement au client le contenu de son cache. Ceci va nous permettre de limiter le trafic extérieur, par conséquent d'améliorer les temps de réponse.

Le transparent proxy fonctionne suivant le même principe, sauf que les requêtes ne lui sont pas transmises directement. Cette configuration utilise le caching proxy ainsi qu'un routeur qui va se charger de rediriger les requêtes du client sur le proxy. Ceci permet notamment de supprimer la configuration de chaque client afin que ceux-ci utilisent le serveur proxy.



2.2.Reverse proxy (IP forwarding)

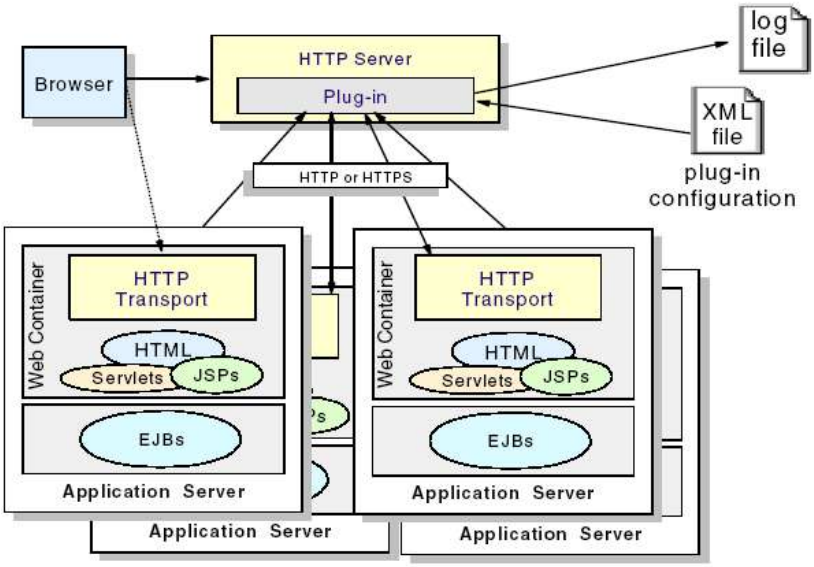


Le reverse proxy permet lui aussi de limiter la charge de d'informations au niveau des serveurs de l'entreprise. Au lieu d'accéder directement au serveur interne de l'entreprise, l'utilisateur va faire ses demandes auprès du reverse

proxy. En fonction de la demande du client, le proxy va soit renvoyer le contenu de son cache si les informations demandées sont en mémoire, soit interroger pour l 'utilisateur le serveur de l'entreprise. Ce mécanisme permet de protéger les serveurs de l'entreprise, mais également de diminuer le taux de charge de ceux-ci.

3. Répartition de charge entre le serveur Web et les Container Web

Comme vous avez pu le constater dans les chapitres précédents, la répartition de charge peut se faire aux niveaux des serveurs web de l'entreprise. WebSphere permet également d'effectuer de la répartition de charge entre le serveur web et les différents container Web. Les Web containers ont pour rôle de gérer les composants J2EE Servlet et JSP. Pour effectuer cette répartition de charge, il faut tout d'abord que ces composants soient membre d'un cluster (cellule) et que le serveur web intègre un plugin WebSphere lui permettant d'accéder aux différents membres du cluster.
La figure ci-dessous présente l'architecture mise en place pour la répartition de charge. Dans ce cas, nous avons quatres container web et un serveur web intégrant le plugin. Ce plugin va permettre en fonction des demandes du client (browser) d'orienter les requêtes sur le container web le moins surchargé. Pour cela, il dispose tout d'abord d'un fichier XML lui indiquant les paramètres du cluster, les modules disponible, ainsi que le taux de charge supporté par chacun des membres.

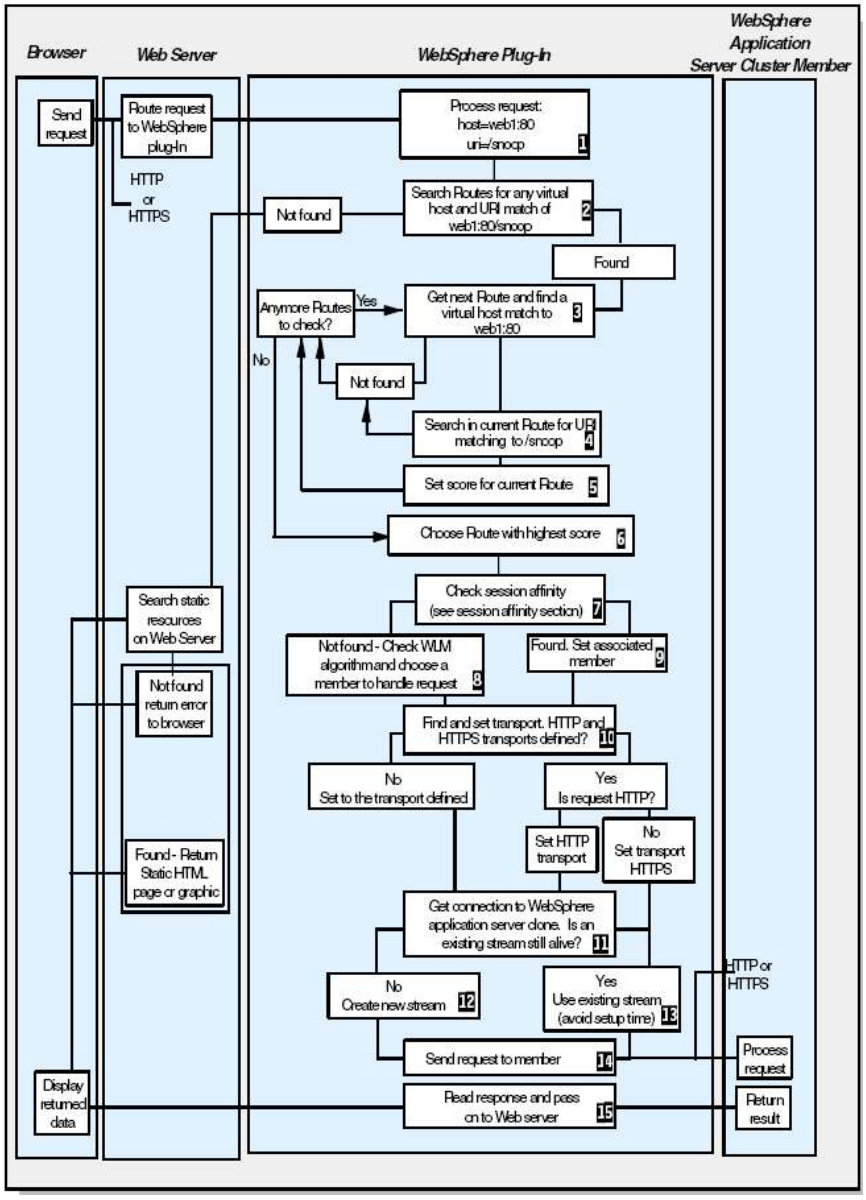


La description du mécanisme qui va suivre est une traduction de la documentation d'IBM. Nous allons décrire chaque boite du schéma suivant:

Dans ce cas de figure, nous supposons qu'un utilisateur veut accéder à la page web http://web1:80/snoop depuis sont navigateur web. Cette requête est routée jusqu'au serveur Web intégrant le plugin de répartition de charge.

1. Le serveur web reçoit la requête et la transmet au plug-in de WebSphere (**1**).
2. Le Plug-in commence par charger à partir de son fichier plugin-cfg.xml toutes les routes définies vers les différents membres du cluster. Il recherche si il connait l'hote, le port et la page demandée grâce au fichier précédemment chargé **2,3,4.**
3. Par rapport aux différentes routes disponibles, il choisit la route ayant le score le plus grand (dépend du fichier de configuration) **5,6**.
4. Ensuite le plugin va vérifier si une session d'identification est déjà établie. (**7,8,9**).
5. Le plugin choisit ensuite le membre du cluster qui va se charger de répondre à la requête.

6. Ensuite, le cluster membre dispose de deux protocoles de transport HTTP ou HTTPS. Comme cette requête est du type HTTP, le membre du cluster utilise le protocole de transport HTTP(**10**).
7. Dans la boite (**11**), on utilise le terme Stream. Le Stream est une connexion au container Web. Depuis la version HTTP 1.1, il est possible de maintenir des connexions entre le plug-in et le container Web. Dans cette exemple, il n'y a pas de connexion établie, donc on crée un nouveau Stream (**12**). Si la connexion était déja établie, le plug-in utilise ce stream (**13**).
8. Enfin, la requête est envoyé sur le cluster membre (**14**). Le plug-in récupère la réponse du container Web, la transmet au serveur Web qui lui la retourne à l'utilisateur.



4. Répartition de charge entre les EJB

4.1. Rappel sur les Entreprise JavaBean

Les entreprises JavaBeans sont un modèle de composants du coté serveur qui permet de développer des objets métier pouvant migrer d'une marque de conteneur d'EJB à une autre. Un bean métier présente un modèle de programmation simple permettant au développeur de se concentrer sur l'aspect métier.

Les composants côté serveur des Entreprise JavaBeans sont de trois types: **les beans entité, les beans session et les beans orientés message**. Les beans session et les beans entité sont des composants coté serveur basés sur RMI (Remote Method Invocation). On y accède à l'aide des protocoles d'objets distribués. Les beans orientés message disponible depuis la version 2.0 sont des composants côté serveur asynchrones qui répondent aux messages JMS(Java Message Service) asynchrones.

- **Les beans entité** modélisent des concepts métier pouvant être désigné par des noms. Par exemple un bean entité peut représenter un client, un objet d'un stock. Autrement dit, les bean entité correspondent à des objets réel. Ces objets sont dans la plupart des cas des enregistrements persistants stockés dans une base de données.
- **Les beans session** sont une extension de l'application client et ils sont responsable de la gestion des processus ou des tâches.
- **Les beans orientés message** des EJB 2.0 sont responsables de la coordinations des tâches impliquant d'autres bean session et entité. La différence entre les beans orientés message et les beans entité réside dans la façon de d'y accéder. Un Bean session fournit une interface distante afin de définir les méthodes que l'on peut invoquer, alors que les Beans orientés message attend des messages asynchrones spécifiques auxquels il répond.

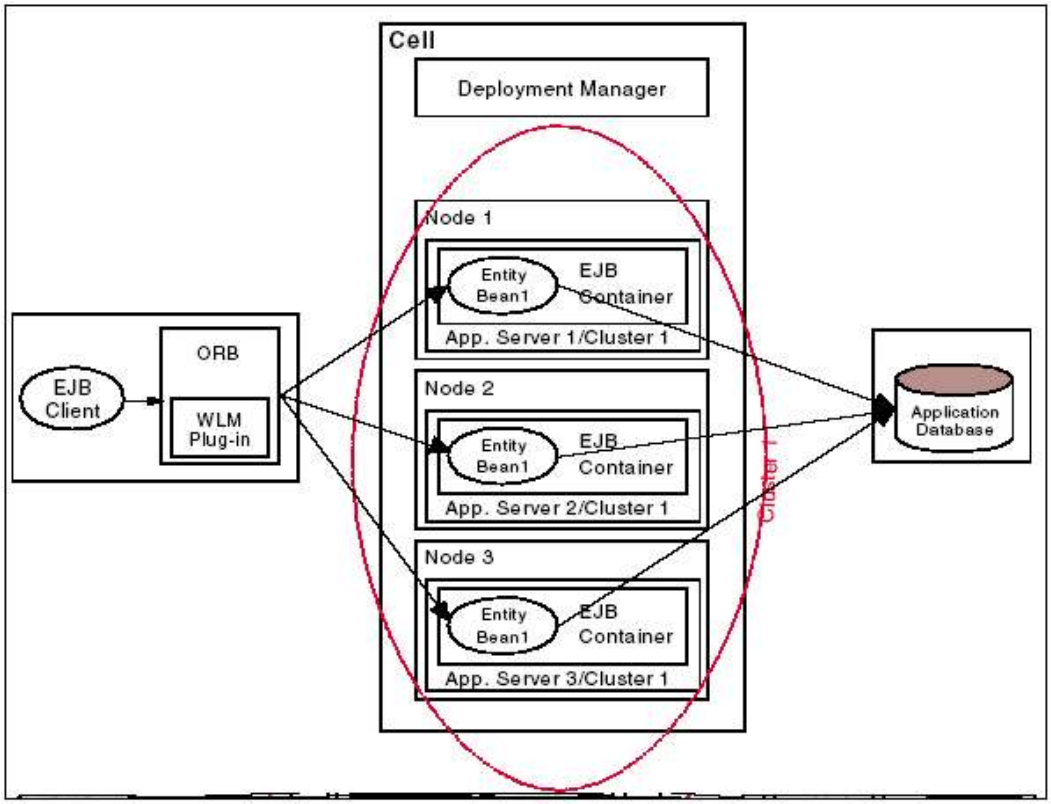
4.2. La répartition de charge EJB sous WebSphere

La répartition de charge est également disponible au niveau des containers EJB. Le principe est le même que pour la répartition de charge au niveau des containers Web. WebSphere met en place un plugin WLM (WorkLoad Management) qui est intégré dans l'ORB (Object request Broker). Cette répartition de charge est opérationnelle dès lors qu'un cluster (Cell) est créé.

Comme vous pouvez le voir sur la figure ci-dessous, nous avons trois serveurs WebSphere qui font partie d'un cluster (Cluster1). Par conséquent, chacun des membres du cluster possèdent des Entreprise JavaBean identiques. Dans ce cas de figure nous avons des Beans de type Entity qui accèdent aux bases de données de l'entreprise.

Avant de comprendre la répartition de charge, il faut savoir que les informations contenues dans un Entity Bean n'est pas associé à une session. Pour les Entity Bean, le concept de session est remplacé par le concept de transaction. Ceci implique que le client accède toujours à la même instance du Bean pendant l'ensemble de la transaction. Ce concept est appelé « transaction affinity ».

Par conséquent, lorsque le client demande un accès à un Bean, le plugin WLM recherche si le client dispose d'une transaction en cours. Si c'est le cas, il est redirigé vers celui-ci. Dans le cas contraire, le plugin dirige le client sur un membre de cluster ayant le taux de charge le plus faible afin d'obtenir une répartition équitable de la charge.


Cell
Deployment Manager
Node 1
Entity Bean1
EJB Container
App. Server 1/Cluster 1
Node 2
Entity Bean1
EJB Container
App. Server 2/Cluster 1
Node 3
Entity Bean1
EJB Container
App. Server 3/Cluster 1
Cluster 1
ORB
EJB Client
WLM Plug-in
Application Database

Description d'autres serveurs d'application

Après avoir étudié WebSphere, nous nous sommes demandé quels étaient les autres solutions disponibles sur le marché. Nous en avons choisi trois que nous allons présenter : BEA WebLogic, Microsoft DNA et Jboss. Afin de pouvoir établir une comparaison entre ces serveurs d'applications, nous avons orienté nos recherches autour de quatre pôles :

- Le moteur du serveurs
- Les outils de développement
- Les fonctionnalités originales du serveur d'applications
- Les outils d'administration

1. BEA WebLogic

WebLogic est un serveur d'application qui est a 100% conforme à la norme J2EE version 1.3. Les caractéristiques de ce serveur sont les suivantes :

- **moteur :** WebLogic Application Server écrit à 100% en JAVA.
- **développement :** Bibliothèque de classe sont fournit permettant d'utiliser n'importe quel outils de développement. Il existe néanmoins un outils proposé par BEA : WebLogic Builder
- **administration** : console d'administration web graphique très claire, simple d'utilisation et avec des possibilités immenses (statistique du serveur visible directement sous forme de différents graphiques, ...)

Du fait que son moteur a été développé à100% en java, WebLogic est très dépendant de la qualité de la JVM. Ainsi de grandes différences de performance peuvent apparaître d'une version a l'autre de la JVM qui sont indépendant de BEA. WebLogic inclus aussi un panel de drivers JDBC natifs qui lui offre une compatibilité et une optimisation avec un très grand nombre de base de données du marché.

Une étude comparative a été mené entre WebLogic et WebSphere qui a montré que WebLogic avait des performances 30% supérieur à WebSphere tout en ayant un coût par transaction (basé sur les performances) inférieur de 60% par rapport à WebSphere.

WebLogic est à l'heure actuelle la meilleure solution de serveur d'applications du marché.

2. Microsoft DNA (Distributed interNet Application)

Microsoft DNA n'est pas un serveur d'applications. Nous avons décider de le présenter car le concept de Microsoft DNA est totalement dans la philosophie des serveurs d'applications et régie l'ensemble des solutions actuellement vendues par Microsoft.

En effet Windows DNA définit une structure de développement de solutions répondant aux besoins exigeants de l'informatique professionnelle, d'Internet, des réseaux intranet et du commerce électronique global, tout en réduisant les coûts globaux de développement et de mise en œuvre.

Cette structure de développement repose sur sur les composant suivants :

- **moteur :** MTS (Microsoft Transaction Server) et IIS (Internet Information System)
- **développement :** la suite VisualStudio
- **administration** : L'administration se fait par les outils disponible dans les version de Windows NT.

L'architecture Microsoft DNA utilise les services Windows standard pour répondre aux différentes exigences de la solution à déployer : interface utilisateur et navigation, logique et stockage des données. Les services utilisés dans Microsoft DNA, qui sont intégrés par l'intermédiaire du modèle COM (Component Object Model), sont les suivants :

- Dynamic HTML (DHTML)
- Active Server Pages (ASP)
- Composants COM
- Services de composants
- Services Active Directory
- Services de sécurité Windows®
- Service de mise en file d'attente de messages Microsoft®
- Composants Microsoft Data Access

Un des gros point faible des solutions reposant sur le serveur transactionnel MTS est le module de répartition de charges. En effet comme nous l'avons vu avec WebSphere, il existe différentes manière de gérer la répartition de charge. Microsoft a fait le choix de la répartition de charges statique.

La grande question est « qu'est ce que la répartition de charges statique ? » En effet cette politique est une forme rudimentaire de répartition de charges qui se base sur les ressources estimées disponibles. Supposons que deux serveurs MTS sont disponibles pour une même application serveur. Alors, pour permettre une répartition des charges entre ces deux serveurs, il faudra dimensionner la capacité de chaques serveurs en se basant sur la capacité estimée de chacun des serveurs (dans notre exemple, le premier serveur peut supporter la charge de 10 clients simultanés, le second 100). Les clients à leurs connexions seront redirigés vers les serveurs dans leurs ordres de configuration (dans l'exemple 11 client se connecte à l'application, les dix premiers seront pris en charges par le premier serveur, le dernier par le deuxième serveur). Cette solution n'est clairement pas optimisée, car si le premier serveur a un problème quelconque alors la plupart des clients seront impactés par les problèmes.

Néanmoins, une forme de répartition dynamique de charges est a l'étude chez Microsoft et devrait être intégrée dans une prochaine version de MTS.

3. JBoss

JBoss est un serveur d'applications open-source J2EE (distribué sous licence LGPL), actuellement a la version 3.2 pour des serveurs de production et a la version 4.0 pour les développements. Jboss est écrit à 100% en JAVA. A première vue, on pourrait se dire « projet open source pas de garantie de continuité » mais pourtant Jboss est un projet très actif et qui est soutenue par une société de services qui fournit essentiellement des formations à Jboss et des développement de services sur Jboss.

Les caractéristiques du serveur d'application Jboss sont les suivantes :

- **moteur :** basé sur JMX (Java Management eXtension)
- **développement :** Via un plugin pour l'environnement « Eclipse » d'IBM
- **administration** : interface web pas très conviviale (illisible)

D'un point de vue structurel, Jboss est un rassemblement d'implémentation de la norme J2EE autour d'un noyau basé sur JMX. Les modules de Jboss 3.0 sont les suivants :

- **JBossServer** qui comporte une infrastructure constituée des conteneurs EJB, ainsi que du Java Management Extension (JMX). Ce composant constitue le coeur du serveur d'application Jboss. Son rôle est de gérer le noyau grâce au JMX, mais également de fournir les conteneurs EJB. Contrairement à la plupart des serveurs d'applications J2EE, JbossServer permet de déployer et redéployer automatiquement les nouvelles applications sans avoir à stopper ou arrêter le serveur. Comme vous pourrez le voir dans l'installation du serveur Jboss, un répertoire nommé deploy permet de déployer les nouvelles applications de manière automatique.
- **JBossMQ** pour la gestion des messages JMS (Java Messaging service). Ce composant est apparu en avril 2000. Il implémente l'API Java Messaging Service (JMS). Le JMS permet un échange entre des applications ou des composants via des brokers de messages ou MOM (Middleware Oriented Messages). Ces échanges peuvent se faire dans un contexte interne (pour l'EAI) ou un contexte externe (pour le B2B).
- **JBossTX** pour la gestion des transactions avec les API JTA(Java Transaction API) et JTS(Java Transaction Service). Ce composant permet le support des moniteurs de transaction avec les API JTA/JTS. Le JTS définit l'implémentation du manager de transaction qui supporte l'API de transaction JTA. Malgré que Jboss implémente la manager de transaction JTS, vous ne pouvez pas accéder directement aux méthodes. Vous devez utiliser l'API JTA pour invoquer les routines de bas niveau du JTS.
- **JBossCMP** pour la persistance CMP. Ce module permet de gérer la connexion au base de données grâce au connecteur JDBC. La version actuelle de JBOssCMP permet de se connecter à 17 bases de données (Oracle, SQLServer, DB2, Sybase, PointBase, Cloupscape, HypersonicsSQL, PostgreSQL, mySQL, etc ...)
- **JBossSX** pour la sécurité basée sur JAAS (Java Authentication and Authorization Service). Ce composant a pour rôle de gérer la sécurité grâce à l'API JAAS. JbossSX fournit une implémentation de la sécurité standard à J2EE. Il permet notament l'authentification des utilisateurs grâce au module JAAS Login.
- **JBossCX** pour la gestion des connecteurs avec JCA (J2EE Connector Architecture). JbossCX permet de gérer les connecteurs aux systèmes d'informations de l'entreprise comme CICS, TUXEDO, SAP, Siebel, etc...
- **Tomcat** ou **Jetty** pour le support des servlets et des pages JSP.

Même si Jboss est un produit open source, il n'en est pas moins évolué. En effet avec ses implémentations de la norme J2EE, il acquiert une architecture du serveur et de sécurité qui pourrait venir concurrencer les leaders du marché des serveur d'applications.

Conclusion

WebSphere est donc un serveur d'application particulièrement performant d'IBM. En effet même si c'est un produit encore relativement jeune (vis a vis du leader Weblogic), il peut se targuer de posséder des fonctionnalités aussi avancés que WebLogic tout en innovant certaine implémentation de la norme J2EE.

Néanmoins, WebSphere est un serveur d'application lourd. Gourmand en ressource, il n'est pas compatible tout système comme il est pourtant annoncé par IBM. En effet si la configuration est un peu différente de celle annoncée (exemple typique : les « locals » en français) alors il peut y avoir des dysfonctionnements et des incompatibilités entre composants internes.

Enfin nous avons trouvé aussi très intéressant les nombreux outils de développements proposés par IBM ainsi que la facilité d'obtention d'une licence spéciale de la part du service commercial d'IBM pour faire mettre en place une démonstration sur le LoadBalancing qui n'a malheureusement pas fonctionné.

Nous tenons également à remercier Monsieur MOURIER pour nous avoir fourni la documentation nécessaire à la compréhension de ce serveur.

Références

Voici les références qui ont été consulté lors de la rédaction de ce rapport ainsi que pour la préparation de l'exposé:

- http://www.redbooks.ibm.com/abstracts/sg246573.html :
  -Tous les aspects de la sécurité de la version 5.0 de WebSphere

- http://fr.bea.com/index.jsp

- http://www.jboss.org/

- http://www.bea.com/content/news_events/white_papers/BEA_WLS_vs_Websphere_TCO_wp.pdf :
  - Comparatif WebLogic / WebSphere

- **Serveurs d'applications** Thierry Brethes - Francois Hisquin - Pierre Pezziardi edition Eyrolles :
  Bonne présentation des serveurs d'applications du marché par contre devient un peu vieux vu que cette édition date de l'an 2000

FIN DU DOCUMENT