République Algérienne Démocratique et Populaire

Université Saad DAHLEB de BLIDA

Faculté des Sciences

Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

OPTION : SYSTEMES INFORMATIQUES ET RESEAUX (S.I.R)

Thème

Etude et Réalisation d'une Plate-forme Hautement Disponible

Réalisé par :

- ***Mr. DERMIME Seyf Eddine***

Présenté le 23 octobre 2019 devant le jury composé de MM.

- *Mr. OULD AISSA* ***(Président)***
- *Mr. BENYAHIA Mohamed* ***(Promoteur)***
- *Mr. BELKHODJA Yacine* ***(Encadreur)***
- *Mr. KEZZAR Khalid* ***(Encadreur)***
- *Mr. BEY* ***(Examinatrice)***

Année Universitaire: 2018-2019

Remerciements

Nous remercions avant toute chose dieu le tout puissant qui a guidé nos pas pour l'accomplissement de ce modeste travail.

Nous tenons aussi à remercier et à exprimer notre profonde gratitude à Dr « BENYAHIA », notre promoteur de nous avoir fait confiance durant le projet.

Nous remercions notres encadreurs de stage Mrs « BELKHODJA, KEZZAR» pour le temps précieux qu'ils nous a accordé.

Nous adressons aussi nos remerciements au président et aux membres du jury qui nous font honneur en acceptant de juger notre travail.

Notre reconnaissance s'adresse à nos familles qui ont su nous apporter, sans relâche, leurs soutiens durant toutes ces longues années d'études.

Enfin que tous ceux qui, de prés ou de loin ont contribué à l'aboutissement de ce travail soient assurés de nos profondes gratitudes.

Résumé :

Produire des systèmes stables demande de passer beaucoup de temps en études et en analyse. Il existe des techniques simples permettant de pallier à la fiabilité des systèmes complexes, qu'ils soient matériels ou logiciels.

Si l'on peut prévoir la panne d'un composant matériel ou logiciel, on peut alors l'anticiper et mettre en œuvre une solution de substitution. On parle alors de disponibilité du service, voire de haute disponibilité selon la nature de l'architecture mise en place.

L'objectif de la présente étude consiste à présenter une solution de haute disponibilité au niveau de la plateforme du Progiciel de Gestion pour la Recherche et l'Enseignement Supérieur (PROGRES) du Ministère de l'Enseignement Supérieur et de la Recherche **S**cientifique (MESRS) en utilisant la solution HA (Haute Disponibilité), cette solution consiste à mettre en place une nouvelle architecture.

Une architecture hiérarchique interconnectant différent serveurs (Cluster) proposée afin de mettre en place une infrastructure hautement disponible pour améliorer la capacité du service en perpétuelle croissance, et assurer la disponibilité du service, et réduire au maximum les risques d'incidents.

Mots clés : Haute Disponibilité, Capacité, Cluster.

Abstract:

Producing stable systems requires spending a lot of time studying and analysing. There are simple techniques to overcome the reliability of complex systems, whether hardware or software.

If we can predict the failure of a hardware or software component, we can then anticipate and implement a substitution solution. We then talk about availability of the service, or even high availability depending on the nature of the architecture put in place.

The objective of this study is to present a high availability solution at the platform of the Management Software Package for Research and Higher Education (PROGRES) of the Ministry of Higher Education and Scientific Research (MESRS) using the HA (High Availability) solution, this solution consists in setting up a new architecture.

A hierarchical architecture interconnecting different servers (Cluster) proposed to implement a highly available infrastructure to improve the capacity of the service in perpetual growth, and ensure the availability of the service, and minimize the risk of incidents.

Keywords: High Availability, Capacity, Cluster.

ملخص :

يتطلب إنتاج أنظمة مستقرة قضاء الكثير من الوقت في الدراسة والتحليل. هناك تقنيات بسيطة للتغلب على موثوقية الأنظمة المعقدة ، سواء كانت أجهزة أو برامج.

إذا استطعنا التنبؤ بفشل أحد مكونات الأجهزة أو البرامج، فيمكننا عندئذٍ توقع وتنفيذ حل بديل. نتحدث بعد ذلك عن مدى توفر الخدمة ، أو حتى مدى التوفر الكبير اعتمادًا على طبيعة البنية المعمول بها.

الهدف من هذه الدراسة هو تقديم حل عالي التوفر على منصة حزمة البرامج الإدارية للبحوث والتعليم العالي (PROGRES) التابعة لوزارة التعليم العالي والبحث العلمي (MESRS) باستخدام حل التوافر العالي (Haute Disponibilité)، يتكون هذا الحل في إنشاء بنية جديدة.

اقترحت بنية هرمية تربط خوادم مختلفة (Cluster) بنية تحتية متوفرة للغاية لتحسين قدرة الخدمة في النمو الدائم ، وضمان توافر الخدمة ، وتقليل مخاطر الحوادث.

الكلمات الرئيسية : التوافر العالي ، السعة ، مجموعة خوادم.

Table des matières

Table des matières

Table des matières

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

-C-

CERIST : Centre de Recherche sur l'Information Scientifique et Technique.

Cluster : Une grappe d'ordinateurs.

-D-

DMZ : Demilitarized Zone.

DNS : Domain Name System.

DRBD: Distributed Replicated Block Device.

DRSICU : Direction des Réseaux et Systèmes d'Information et de Communication Universitaires.

DR: Disaster Recovery.

DR-BDD: Disaster Recovery Base De Données.

DR-NFS: Disaster Recovery Network File Système.

DR-APP: Disaster Recovery Application.

-F-

FOS: FailOver Services.

Failover : Basculement automatique.

-G-

GFS : Global File System.

-H-

HA : High Availability (Haute Disponibilité).

HTTP: Hypertext Transfer Protocol, C'est le protocole de transfert sur internet le plus courant.

Heartbeat : Logiciel de surveillance de la disponibilité des programmes sous linix, FreeBSD, solaris, MacOS X. il est sous la licence GPL.

HACMP : HACMP (High-Availability Cluster MultiProcessing) est une technique de haute disponibilité utilisée par IBM, à base de Reliable Scalable Cluster Technology. En 2013, la version 7.1 est disponible pour AIX. La version 5.4 est disponible pour Linux sous licence IBM (le type exact de licence est IPLA).

Liste des abréviations

-I-

IT : Information Technologie.

IP : Internet Protocol.

-J-

JDK : Java Development Kit , désigne un ensemble de bibliothèques logicielles de base du langage de programmation Java, ainsi que les outils avec lesquels le code Java peut être compilé, transformé en bytecode destiné à la machine virtuelle Java.

JSP : Java Server Pages, est une technique basée sur Java qui permet aux développeurs de créer dynamiquement du code HTML, XML ou tout autre type de page web.

-L-

LVS : Linux Virtual Server.

-M-

MESRS : Ministère de l'Enseignement Supérieur et de la Recherche **S**cientifique.

MTBF : Mean Time Between Failure (Temps moyen entre les échecs).

MTTF : Mean Time To Failure (Temps moyen jusqu'à l'échec).

MTTR : Mean Time To Repair (Temps moyen de réparation).

MySQL: est un serveur de bases de données relationnelles Open Source.

Mon : Mon est un outil de monitoring généraliste. Il permet de surveiller des ressources (services réseau, ressources locales ?) et de déclencher des actions selon leur état (lancement de scripts, notification par E-Mail, pager, envoi de *traps* ?)

-N-

NAS: Network Attached Storage

-O-

OSI: Open System Interconnection.

Liste des abréviations

-P-

PGI : Progiciel de Gestion Intégré.

PROGRES : PROgiciel de Gestion pour la Recherche et l'Enseignement Supérieur**.**

-R-

RAID : Redundant Array of Independant Disk. C'est une grappe de disque dur.

RSYNC : Remote Synchronization.

-S-

SDIBR : Sous-Direction de l'Infrastructure de Base et Réseaux.

SGBD : Système de Gestion de Base de Données, est un logiciel système conçu pour créer et gérer des bases de données.

SAN : Storage Area Network.

SSL : Secure Socket Layer.

-T-

TCP : Transmission Control Protocol.

Trucluster : est une solution de clustering haute disponibilité à source fermée pour le système d' exploitation Tru64 UNIX . Développé à l'origine par Digital Equipment Corporation (DEC), il a ensuite été transféré à Compaq en 1998 lors de l'acquisition de Digital par la société, qui a ensuite fusionné avec Hewlett-Packard (HP).

-V-

VLAN: Virtual Local Area Network.

VPN : Virtual Private Network.

VIP : Virtual IP

-X-

XML: eXtensible Markup Language, est un langage informatique qui sert à enregistrer des données textuelles.

Introduction générale

Produire des systèmes stables demande de passer beaucoup de temps en études et en analyse. Il existe des techniques simples permettant de pallier à la fiabilité des systèmes complexes, qu'ils soient matériels ou logiciels. Plutôt que de chercher à rendre ces systèmes stables, on peut inverser la démarche et intégrer à la source la notion de panne dans l'étude de ces systèmes : si l'on peut prévoir la panne d'un composant matériel ou logiciel, on peut alors l'anticiper et mettre en œuvre une solution de substitution. On parle alors de disponibilité du service, voire de Haute Disponibilité selon la nature de l'architecture mise en place.

la DRSI constitue une direction centrale au niveau du ministère et qui a un rôle, d'une part, de fournisseur des services informatique destinés aux différentes structures centralisée et décentralisée tel que les universités, les grandes écoles rattachées à ce dernier, et de soutenir tous les services des IT et les autres infrastructures. Il est cependant évident qu'un ordonnancement des services soit établi selon le niveau de critique des activités métier couvertes par les services informatiques. Il est donc nécessaire que les services soient aussi à leurs tours classées par niveau de critique correspondant à celui de l'activité métier. Ces derniers seront sujets à des prérequis nécessaires en termes de Disponibilité et de Continuité. C'est pourquoi le problème de haute disponibilité est posé et doit avoir une réponse par la réalisation de ce projet.

Dans le cadre du programme gouvernemental de modernisation et restructuration du système d'information et consécutivement à l'évolution technologique de ces dernières années, ainsi que la montée en puissance des demandes de services informatique en termes de fourniture d'applications, d'informations, de service et de support technique , par conséquent la Direction des Réseaux et Systèmes d'Information (DRSI) du MESRS a inscrit le projet ambitieux permettant de répondre aux besoins et de mettre en place une plateforme d'inscription pour le Master et le Doctorat et devra répondre à plusieurs contraintes tels que le temps et la prise en charge des informations .

Introduction générale

Le niveau de critique est lié à l'application PROGRES par rapport à la date de début et fin d'inscription, les informations traitées pour les dossiers d'inscription (Master et Doctorat) et dans un futur proche les nouveaux bacheliers jusqu'à la prise en charge de toutes les années universitaire.

D'année en année on remarque une énorme évolution du nombre des étudiants et des établissements universitaires sur le territoire national donc un nombre important de places pédagogiques par conséquent un volume important d'informations à traiter ce qui nécessite une architecture et infrastructure réseau pouvant prendre en charge de l'acquisition, le traitement et le stockage de ces information avec les critères dont elles font l'objet, c'est-à-dire la disponibilité , la fiabilité et la capacité . Vu l'importance des informations qui sont souvent véhiculées dans les réseaux, ceci requièrent une bonne gestion du réseau, une souplesse d'utilisation et un certain degré de sécurité. Ces derniers sont devenus des éléments clés de la continuité des systèmes d'information.

D'autre part cette nouvelle plateforme d'inscription règle la politique de proximité Université-Etudiant ce qui permet à l'étudiant de s'inscrire depuis chez lui via la plateforme PROGRES qui est accessible sur le web, cela permet d'éviter aux étudiants de parcourir des dizaines, voire des centaines de kilomètres pour s'inscrire et refaire la même chose lors de l'affichage des résultats.

Les incidents finissent toujours par arriver, le risque zéro n'existe pas. Le MESRS doit prendre toutes les mesures visant à garantir la continuité de service du système informatique. Il s'agit de maintenir l'informatique d'entreprise en bon état de fonctionnement en limitant la fréquence et la durée des interruptions. La haute disponibilité permet à l'ensemble du système informatique de continuer à fonctionner malgré la défaillance d'un ou plusieurs éléments matériels ou logiciels.

Introduction générale

C'est pourquoi, il est temps de revoir toute l'architecture des systèmes et l'infrastructure de services afin de trouver la meilleure solution qui répond et répondra aux demandes changeantes des services actuelles et futures et d'assurer les exigences légales, contractuelles et les exigences des niveaux de service.

Dans ce projet on réalisera un volet très important et qui réside à mettre en place une infrastructure de services hautement disponible et règlera nécessairement deux exigences importantes (capacité et disponibilité des services) et impactera positivement les opérations et le support (moins d'incidents relatifs à la disponibilité et à la capacité).

Nous commençons notre mémoire par une introduction générale.

Le mémoire est organisé en trois chapitres :

La première chapitre : Plateforme hautement disponible (Etat de l'art).

Ce chapitre décrit comment rendre une plateforme hautement disponible (les approches et les logiciels).

Le deuxième chapitre : Conception de la solution.

Ce chapitre décrit l'étude de l'existant et la conception de notre solution (nous présenterons l'architecture technique de notre solution).

Le troisième chapitre : Réalisation.

Dans ce chapitre, nous présenterons l'ensemble des outils utilisés (Matériels et Logiciels) ainsi que les processus de la configuration implémentée de la plateforme.

Nous achèverons notre mémoire par une conclusion générale.

Chapitre 1 : Plateformes hautement disponible (Etat de l'art)

Introduction :

On parle dans ce chapitre à l'état de l'art ou comment rendre une plateforme hautement disponible avec les déférences approches et les déférents logiciels existés.

1. Définitions de la haute disponibilité :

- La haute disponibilité est un terme souvent utilisé en informatique, à propos d'architecture de système ou d'un service pour désigner le fait que cette architecture ou ce service a un taux de disponibilité convenable, cette dernière concerne de plus en plus d'entreprises comme de particuliers. On appelle haute disponibilité (High Availability) toutes les dispositions visant à garantir la disponibilité d'un service, c'est-à-dire assurer le bon fonctionnement de ce dernier.

- La haute disponibilité (ou High Availability ou HA) permet d'assurer et de garantir le bon fonctionnement des services ou applications proposées et ce 7j/7 et 24h/24. Cela consiste donc à mettre en place toutes les actions et dispositions techniques pour qu'une infrastructure informatique soit toujours disponible en appliquant certains principes tels que la réplication des données, la sauvegarde, la répartition de la charge, la redondance, etc. pour limiter l'indisponibilité d'un SI.

- Ainsi, un cluster de haute disponibilité en anglais « High Availbility », est un ensemble de serveurs physiques, au nombre minimum de deux, afin d'obtenir une activité de services en temps réel, en toutes conditions, de l'ordre de 99.99%.
  Ainsi, dans une architecture à haut risque où les services doivent être disponibles 24 heures sur 24, 7 jours sur 7, une solution devrait être en mesure d'assurer cette disponibilité. [1]

2. Fiabilité VS Disponibilité :

Pour appréhender la notion de Haute Disponibilité, il nous faut d'abord aborder les différences qui existent entre la fiabilité d'un système et sa disponibilité.

La fiabilité est un attribut permettant de mesurer la continuité d'un service en l'absence de panne. Les constructeurs fournissent généralement une estimation statistique de cette valeur pour leurs équipements : On parle alors de MTBF (pour Mean Time Between Failure). Un MTBF fort donne une indication précieuse sur la capacité d'un composant à ne pas tomber en panne trop souvent. Dans le cas d'un système complexe (que l'on peut décomposer en un certain nombre de composants matériels ou logiciels), on va alors parler de MTTF pour Mean Time To Failure, soit le temps moyen passé jusqu'à l'arrêt de service consécutif à la panne d'un composant ou d'un logiciel. [1]

La disponibilité en ce qui la concerne, est plus difficile à calculer car englobant la capacité du système complexe à réagir correctement en cas de panne pour redémarrer le service le plus rapidement possible. Il est alors nécessaire de quantifier l'intervalle moyen de temps ou le service est indisponible avant son rétablissement : On utilise l'acronyme MTTR (Mean Time To Repair) pour représenter cette valeur.

Ainsi, un système qui aspire à une forte disponibilité se doit d'avoir soit un MTTF fort, soit un MTTR faible.

Afin de calculer la disponibilité, les métriques suivantes sont utilisées:

- ✓ MTBF (Mean Time Between Failure) : mesure du temps estimé entre deux défaillances d'un système.
- ✓ MTTR (Mean Time to Resolution) : mesure du temps estimé pour restaurer la fonctionnalité.

La formule de calcul de disponibilité est :

Disponibilité = MTBF / (MTBF + MTTR) [2]

Une autre approche, plus pratique, consiste à mesurer la période de temps ou le service n'est plus rendu pour évaluer le niveau de disponibilité. C'est la méthode la plus souvent utilisée même si elle ne tient pas compte de la fréquence des pannes mais plutôt de leur durée.

Le calcul se fait le plus souvent sur une année calendaire. Plus le pourcentage de disponibilité du service est fort, plus nous pouvons parler de Haute Disponibilité.

Type de système	Indisponibilité (minutes par an)	Disponibilité (%)	Classe de disponibilité
Non géré (Unmanaged)	50000	90	1
Géré (Managed)	5000	99	2
Bien géré (Well Managed)	500	99,9	3
Tolérant les fautes (Fault-tolerant)	50	99,99	4
Haute disponibilité (High Availability)	**5**	**99,999**	**5**
Très haute disponibilité (Very High Availability)	0.5	99,9999	6
Ultra haute disponibilité (Ultra High Availability)	0.05	99,99999	7

Tableau 1 : Classification des systèmes en fonction de leur disponibilité. [3]

3. Condition de mise en place :

Une bonne démarche, permettant de mettre une œuvre assez rapidement une infrastructure solide capable de garantir la Haute Disponibilité d'un service critique, consiste à évaluer les différents objectifs suivants :

- Définition des critères d'indisponibilité du service : Niveau de disponibilité, temps de rétablissement ou encore temps d'engagement du service.
- Analyse de la volumétrie des données et des performances nécessaires au bon fonctionnement du service.

- Prise en compte des différents critères de coûts.
- solution des configurations matérielles et logicielles.
- Surveillance du service et planification de la maintenance corrective et préventive (qui, quand, comment).

La haute disponibilité, comment ça marche ?

Pour améliorer la haute disponibilité, il existe de nombreuses solutions correspondant à différents domaines, comme :

- ✓ la redondance de services installés sur des serveurs différents et le basculement d'un serveur à l'autre.
- ✓ la redondance des composants des serveurs ou des éléments d'interconnexion.
- ✓ la répartition dynamique des données sur plusieurs disques durs (RAID, NAS, SAN…).
- ✓ le stockage des sauvegardes à un emplacement géographique différent.
- ✓ les plans de secours.
- ✓ Etc…

4. Approches de la haute disponibilité:

Les supports de Haute Disponibilité existe depuis déjà quelques années sous Linux et même si le niveau de maturité n'est pas encore celui d'autres environnements propriétaires de type Unix, il est déjà largement suffisant dans la plupart des cas.

4.1. La multiplication des ressources critiques (matériels ou logiciels) du serveur :

Une bonne part des techniques disponibles reposes sur la multiplication des ressources critique (physiques ou logicielles) constituant un serveur. En multipliant les ressources, on supprime du même coup leurs caractères critiques. Le service pourra donc être assuré même

en cas de panne d'un composant. Cela permet notamment d'utiliser des composants moins chers puisque la fiabilité du composant ne devient plus le critère principal.

En premier point, l'on devrait tout d'abord définir notre champs d'activité tous les composants d'un système informatique pouvant être d'une manière ou d'une autre mis en haute disponibilité c'est ainsi que, nous nous limiterons à une étude relative aux défaillances logiques (logiciels), et physiques (Terminaux informatique).

❖ Haute Disponibilité au niveau physique (les composons) :

La multiplication des différents composants critiques présents dans notre système peut vous permettre de survivre à une panne en considérant que les solutions matérielles de Haute Disponibilité disponibles sous Linux se rapprochent de plus en plus de celles proposées sur les serveurs Unix haut de gamme.

Il est assez simple de redonder les alimentations électriques (transparent pour les systèmes d'exploitation), mais aussi les disques durs, les contrôleurs disques et les interfaces réseaux :

✓ Alimentation redondée :

Certains constructeurs proposent de fournir deux ou trois alimentations pour prévenir la perte de ce composant. Les alimentations sont des composants critiques et il n'est pas rare de voir celles-ci faillir bien avant les autres composants du système. [1]

✓ Utilisation des grappes de disques :

L'utilisation des technologies RAID est un bon moyen de sécuriser vos données et prendre en compte notamment la perte d'un disque. On peut disposer de cette technologie de façon logicielle sous Linux même car il y est tout à fait envisageable de l'utiliser par l'intermédiaire de cartes d'interface IDE ou SCSI à enficher dans votre système (Linux en supporte un grand nombre). [1]

✓ Multiplication des cartes réseaux :

Un câble réseau peut être accidentellement débranché, Une carte réseau peut subir les aléas d'une panne et ne plus pouvoir être utilisable.

Le service réseau que vous proposez est donc fortement tributaire de la disponibilité de ce type de composants. [1]

- ✓ **Sécuriser l'accès aux unités de stockages externes :**

Le RAID logiciel sous Linux (driver MD) supporte depuis peu un nouveau mode dit Multipath qui s'utilise dans le cas où vous disposez de deux liens physiques qui pointent vers une seule et même ressource.

Un seul des deux liens sera effectivement utilisé et en cas de panne, c'est le second qui prendra la relève. Cela peut être très utile si vous disposez d'une baie de stockage externe en SCSI ou en Fibre Channel disposant de deux interfaces d'entrées/sorties.

Il vous faudra prévoir deux contrôleurs dans votre serveur mais cela reste la solution idéale pour sécuriser vos écritures disque si votre baie de disque supporte cette fonctionnalité. [1]

4.2. La duplication du serveur (Cluster) :

Une seconde approche considère que l'on peut assez facilement mettre en place une solution où ce n'est plus la ressource que l'on va chercher à dupliquer, mais directement le serveur. L'utilisation de grappes de machines (cluster) est un bon moyen de répondre à cette problématique. Si l'on parvient à disposer d'au moins deux machines sur lesquelles le service est exécuté de façon unique (sur l'un ou l'autre des nœuds), la continuité du service sera garantie moyennant le temps de basculement d'une machine à l'autre (**FailOver Services, FOS**).

La principale difficulté consiste à maintenir une copie des données entre les nœuds (dans ce type de cluster dit de Haute Disponibilité, une machine s'appelle un nœud) pour que le service puisse être indifféremment lancé sur l'un ou l'autre des serveurs.

Pour accomplir cela, il existe différentes techniques basées soit sur la réplication plus ou moins en temps réel des données entre les nœuds, soit sur le partage d'une ressource unique en utilisant notamment un système de fichiers distribués ou partagés.

Dans ce type de configuration, il est important de faire en sorte que le temps de rétablissement du service soit le plus faible possible pour réduire le gène occasionné aux utilisateurs.

Le basculement du service dans le cluster ne doit pas être (trop) perceptible et ne doit surtout pas occasionner une modification du paramétrage côté client.

4.3. La répartition de la charge :

Une dernière technique moins connue permet de répartir la charge sur un ensemble de nœuds physiques sur lesquels un service de type réseau est exécuté en parallèle et en concurrence. Un nœud maître se charge de répartir les requêtes sur le nœud le moins chargé du cluster.

Si un nœud tombe en panne, il sera détecté par le maître qui pourra facilement le retirer de sa liste des nœuds actifs.

❖ **Balance de charge** :

La balance de charge permet de répartir l'effort sur plusieurs machines rendant le même service (cluster).

Cette balance peut se faire de plusieurs façons :

✓ **RR (Round-Robin):**

C'est le plus communément utilisé, son principe consiste à distribuer les requêtes sur les différents serveurs un à un en rebouclant. Le RR standard fonctionne de façon « automatique » en adressant chaque serveur à tour de rôle à la chaîne, mais il existe une variante du RR à laquelle on peut ajouter des pondérations dans le cas de serveurs hétérogènes afin de bénéficier au maximum des performances de chaque nœud du cluster. [6]

Les serveurs sont choisis à tour de rôle de manière cyclique.

✓ **Weighted Round Robin :**

Les serveurs sont choisis à tour de rôle, mais relativement à leur poids, c'est-à-dire que les serveurs ayant les poids les plus forts seront désignés en premier et recevront plus de requêtes par rapport à ceux ayant des poids plus faibles. [6]

✓ **Least-Connection :**

Il consiste à déterminer le serveur qui a été le plus anciennement utilisé et avec le plus petit nombre de connexions actives afin de lui transmettre la requête.

Le serveur ayant le moins de connexions (donc le moins chargé) est choisi. [6]

- ✓ **Weighted Least Connection :**

  Le serveur ayant le moins de connexions relativement à son poids est choisi, c'est-à-dire que celui qui minimise le rapport Ci/Wi est désigné (Avec Ci le nombre de connexions du serveur i et Wi son poids). [6]

- ✓ **Destination / Source Hashing :**

  Se base sur l'adresse IP source du client hachée pour déterminer le serveur auquel envoyer la requête. De ce fait, un client retombera toujours sur le même serveur qui pourrait être une solution quant à la persistance de sessions (mais assez limitée). [6]

- ✓ **URI :**

  Le serveur sélectionné dépend directement de l'URI de la requête HTTP. Dans ce cas, on calcule le haché de l'URI (deux cas possibles : la partie gauche de l'URI uniquement (avant le point d'interrogation, c'est-à-dire avant le passage de paramètres) ou alors l'URI entière si elle contient tous les paramètres).

  Ce haché nous permettra de définir le serveur vers lequel envoyer notre requête. Cet algorithme est généralement utilisé dans le cas de proxy de cache dans le but de maximiser le nombre de « hits » (accès au cache et obtention de la donnée avec succès). [6]

- ✓ **HDR :**

Se base sur un champ spécifique de l'entête HTTP afin de déterminer le serveur destination. [6]

- ✓ **Locality-based least connection :**

  Un groupe d'utilisateurs est affecté à chaque serveur (il est choisi en utilisant l'algorithme du weighted least connection), et si celui-ci n'est pas surchargé, il recevra les requêtes des clients qui lui sont attachés. Dans le cas contraire, le serveur qui a un nombre de connexions inférieur à la moitié de son poids est désigné. [6]

- ✓ **Locality-Based least connection with replication :**

  Cet algorithme suit le même principe que celui du locality-based least connection, sauf que les groupes d'utilisateurs sont affectés non pas à un seul serveur, mais à un

ensemble de serveurs, et dans cet ensemble, un serveur est désigné en utilisant l'algorithme du weighted least connection. [6]

- ✓ **Shortest Expected Delay :**

  Le serveur ayant le délai le plus court (estimé selon une formule basée sur le nombre de connexions et le poids) est choisi. [6]

- ✓ **Never queue :**

  S'il y a un serveur idle (au repos), il sera choisi, sinon, l'algorithme du shortest expected delay sera utilisé. [6]

La plus grande partie des architectures misent en œuvre pour garantir la disponibilité d'un service dérivant plus ou moins directement de ces **trois approches** :

La **HA** se divise en plusieurs catégories**, la répartition de charge, la tolérance aux pannes et la réplication des données.** Des technologies existent pour mettre facilement en place une architecture hautement disponible.

La haute disponibilité possède deux grands axes : **La disponibilité des services** et **la disponibilité des données.**

Toutes les techniques permettant de venir à bout des désagréments causés par une indisponibilité temporaire ou permanente sont regroupées sous différentes appellations suivant le degré de la réponse qu'elles apportent au problème posé :

5. Disponibilité des services :

Dans le contexte d'un service critique rendu dans le cadre d'une entreprise (serveur) ou vis-à-vis d'une clientèle (site marchand), une panne occasionnant un arrêt du service peut causer un tort considérable entraînant une perte de productivité, voire de confiance du client. Dans tous les cas, cela peut coûter à court ou moyen terme beaucoup d'argent. On base la démarche sur la disponibilité des données pour ensuite fiabiliser par différentes techniques la continuité du service. [1]

Le fait qu'un service soit fonctionnel en temps réel sans aucune interruption traduit sa disponibilité. Son principe est simple : un service, quelle que soit sa machine de référence

ou les données dont il dispose, doit toujours répondre aux clients qui en font la demande. C'est-à-dire que peu importe le serveur qui répond, lorsqu'un client arrive, le service demandé doit satisfaire la demande.

Le serveur répondant est l'un des serveurs du cluster qui est encore en activité. Dans le cas d'un cluster en haute disponibilité sans load balancing, le serveur maître répond à toutes les requêtes sauf si celui-ci est indisponible. Dans ce cas, c'est le ou l'un des serveurs esclaves qui répond.

Pour de la haute disponibilité de services, **plusieurs types de techniques** existent :

Le FailOver Services **(FOS),** le Linux Virtual Server **(LVS), HAproxy,** Heartbeat**,** OpenBSD, Nginx, Squid…

OpenBSD:

C'est un « firewall » open source performant avec une haute disponibilité du pare-feu avec un master et un slave.

Nginx :

Prononcé comme « engine-ex », est un serveur web open-source qui, depuis son succès initial en tant que serveur web, est maintenant aussi utilisé comme reverse proxy, cache HTTP, et load balancer.

Squid :

Un serveur Squid est un serveur mandataire (proxy) et un mandataire inverse conçu pour relayer les protocoles FTP, HTTP, Gopher, et HTTPS. Contrairement aux serveurs proxy classiques, un serveur Squid gère toutes les requêtes en un seul processus d'entrée/sortie asynchrone.

C'est un logiciel libre distribué sous licence GNU GPL.

5.1. Le FailOver Services (FOS) :

Le failover services est un processus de monitoring et de reprise de services pour seulement deux machines : Le serveur maître et le serveur esclave, chacun surveillant l'autre sur deux canaux et types de connectiques différents.

Le FailOver Services peut vérifier tous services utilisant le protocole TCP/IP et commander l'arrêt ou le démarrage de n'importe quels scripts. Ce dernier contrôle aussi l'état réseau des machines : en d'autre terme, le contrôle de l'IP de la machine.

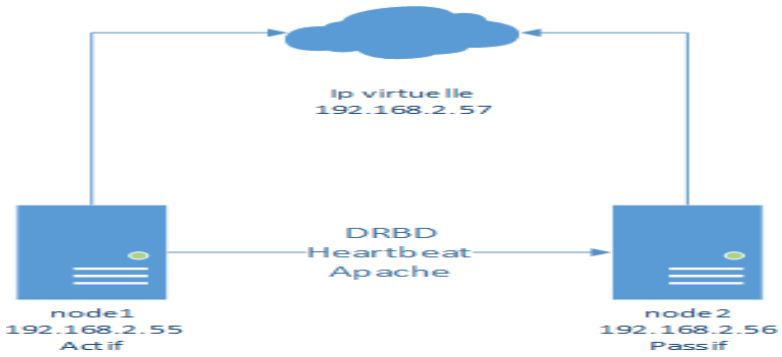


Figure 1 : Schéma montrant l'interconnexion des serveurs pour assurer le service FailOver. [4]

Important : Pour que ce type de haute disponibilité fonctionne, il faut bien sûr que la machine esclave possède les mêmes services que son homologue maitre. Sinon la haute disponibilité ne fonctionnera pas.

5.2. Linux Virtual Server (LVS) :

Linux Virtual Server effectue le même travail que son homologue FOS mais avec un procédé légèrement différent. En effet, LVS s'appuie sur une architecture de Load Balancer et d'un ensemble de serveurs. Donc ce qu'il faut retenir ici est que la haute disponibilité de service se réduit aux trois notions essentielles :

- Le failover (FOS).
- Linux Virtual Server (LVS).
- Et le Load-Balancing (LB).

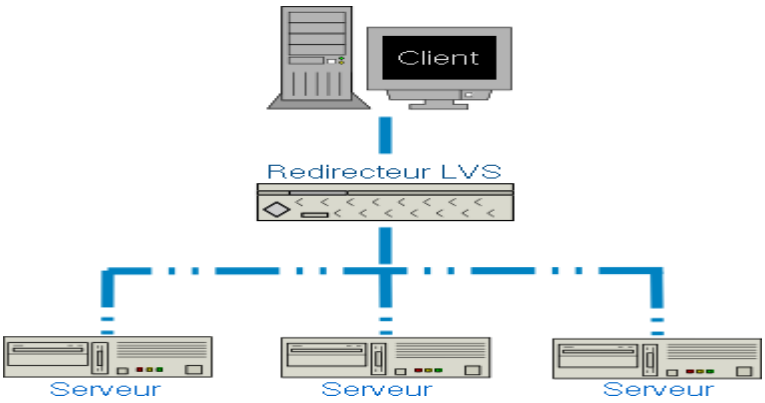


Figure 2 : Schéma montrant un cluster LVS simple. [5]

Le redirecteur LVS utilise des algorithmes de load-balancing pour rediriger les paquets vers les serveurs appropriés.

Au demeurant, disons « qu'un service, quelle que soit sa machine de référence ou les données dont ils disposent, doit toujours répondre aux clients qui en fait la demande quelle que soit les données dont ils disposent ».

5.3. Heartbeat :

Heartbeat est un logiciel de surveillance de la disponibilité des programmes, pour les systèmes d'exploitation Linux, FreeBSD, OpenBSD, Solaris et MacOS X.

Il est distribué sous licence GPL. Heartbeat écoute les battements de cœur, des signaux émis par les services d'une grappe de serveurs lorsqu'ils sont opérationnels.

Lorsque qu'un serveur devient défaillant, Heartbeat le détecte (puisqu'il n'entend plus ses battements de cœurs) et bascule les services surveillés sur un autre serveur. Pour que cela soit transparent pour les utilisateurs, Heartbeat met en place une IP virtuelle unique qui est balancée entre les deux serveurs. [6]

Le trio DRBD - Heartbeat - Apache est parfait pour assurer la redondance de deux machines physiquement proches faisant office des serveurs web.

Assurant à la fois la surveillance des défaillances des systèmes mais aussi celle des services, cette solution met un terme au problème de redondance et d'interruption de service. [6]

5.4. HAProxy :

5.4.1. Description :

HAProxy est une solution open source très flexible offrant des fonctionnalités de répartiteur de charge, reverse proxy HTTP et proxy TCP (c'est-à-dire qu'il fonctionne aussi bien sur la couche 4 que sur la couche 7 du modèle OSI) ainsi que des services de haute disponibilité (HA).

Il est utilisé généralement dans le cas d'un fort trafic et c'est d'ailleurs l'une des raisons pour laquelle il est utilisé par plusieurs grands sites Web, notamment GitHub, Instagram, Twitter, Stack Overflow, Reddit, Tumblr, Yelp et tant d'autres.

5.4.2. Fonctionnalités :

- ✓ Répartition de charge à l'aide de plus de neuf algorithmes d'ordonnancement. Ceci permet de router les requêtes émanant de différents clients vers les serveurs de backend de manière à ce que la charge soit équilibrée au mieux et en évitant toute surcharge sur les serveurs pour permettre un service de qualité rapide et performant. Ceci se fait selon deux modes : le mode TCP et le mode HTTP.
- ✓ Proxy TCP permettant de relayer le trafic entre le client et le serveur en rattachant la connexion ouverte avec le client au serveur pour faire passer le trafic, il joue donc le rôle d'intermédiaire entre les deux.
- ✓ Proxy inverse HTTP, appelé aussi passerelle ou gateway, dans ce cas HAProxy joue le rôle d'un serveur proxy ouvrant deux connexions différentes avec le client et le serveur et fait passer les requêtes entre les deux bouts. L'avantage majeur de ce dernier est de « cacher » des informations sur les serveurs backends.
- ✓ Offre des mécanismes assurant la haute disponibilité afin de garantir la continuité de service et ceci à travers l'utilisation automatique de serveurs répliqués en cas de panne par exemple.
- ✓ Optimisation des ressources et minimisation du temps de réponse.
- ✓ Optimise le trafic et protège les serveurs en évitant de transmettre toute requête invalide aux serveurs.
- ✓ Il offre la possibilité de continuer à fonctionner normalement même en cas de panne des serveurs backend.
- ✓ Permet de faire du monitoring concernant les serveurs et HAProxy lui-même.

- ✓ Offre la possibilité de choisir un serveur bien précis pour lui transmettre la requête et ce, en se basant sur n'importe quel élément figurant dans la requête reçue. Ceci est particulièrement utile pour gérer les sessions des utilisateurs pour qu'un utilisateur soit toujours envoyé vers le même serveur. [6]
- ✓ Il offre des services de health checks permettant de vérifier le bon fonctionnement des serveurs backend ainsi que l'élimination de serveurs défaillants. Ceci se fait à travers un certain nombre de fonctionnalités de vérification offertes par HAProxy. [6]
- ✓ Offre un certain nombre de métriques pouvant être utilisées afin de déterminer les performances et l'état de l'architecture en évaluant certains critères frontend (c'est-à-dire entre HAProxy et les clients) et backend (entre HAProxy et les serveurs), par exemple, il est possible de connaître le nombre de requêtes par seconde, nombre de sessions créées par seconde, nombre d'erreurs HTTP côté client et côté serveur, nombre de tentatives de connexion, etc. Ceci a principalement pour but d'analyser le trafic et prendre des décisions en cas de problème pour améliorer le rendement et les performances.
- ✓ Offre la possibilité d'avoir des connexions chiffrées des deux côtés (client et serveur) avec SSL/TLS. [6]
- ✓ Assure un certain niveau de sécurité en offrant une protection contre les attaques DDoS par exemple en gardant les statistiques de connexions, adresses IP, URL, cookies, etc. pour ensuite appliquer les actions appropriées comme le blocage. [6]

5.4.3. Répartition de charge avec HAProxy :

HAProxy offre des fonctionnalités de répartition de charge (load balancing) assez complètes, dans le sens où elles permettent de réaliser plusieurs configurations personnalisées et adaptées aux besoins de ses utilisateurs. Il existe deux modes de load balancing offerts par HAProxy : le mode **TCP** et le mode **HTTP**.

A. Mode de load balancing (TCP) :

Les décisions de répartition de charge se font sur la base de toute la connexion. Dans ce cas, l'entête de la requête HTTP n'est pas évalué.

B. Mode de load balancing (HTTP) :

Les décisions de répartition de charge se font uniquement sur la base de la requête (chaque requête séparément). Cette méthode permet par exemple de choisir le backend selon l'URL de la requête.

5.4.4. Fonctionnement :

HAProxy considéré comme deux half-proxy, est composé de deux parties essentielles, la partie frontend qui est directement en contact avec le côté client et reste en écoute de ce dernier et la partie backend, du côté des serveurs.

Quand une requête est reçue par le frontend de HAProxy à partir d'un client, celui-ci applique alors les règles définies à ces niveaux tels que le blocage de requêtes, modification des entêtes ou tout simplement l'interception de ces dernières pour établir des statistiques.

Les requêtes sont ensuite envoyées vers le côté backend de HAProxy, qui est relié directement aux serveurs Web où la stratégie de load balancing est appliquée et que se termine par l'envoi de la requête vers le serveur choisi.

Après le traitement de la requête par le serveur, sa réponse est transmise à HAProxy qui peut effectuer quelques traitements dessus ou l'envoyer directement au client via le frontend. Bien évidemment, HAProxy peut être vu comme full-proxy, c'est-à-dire l'union du frontend et du backend.

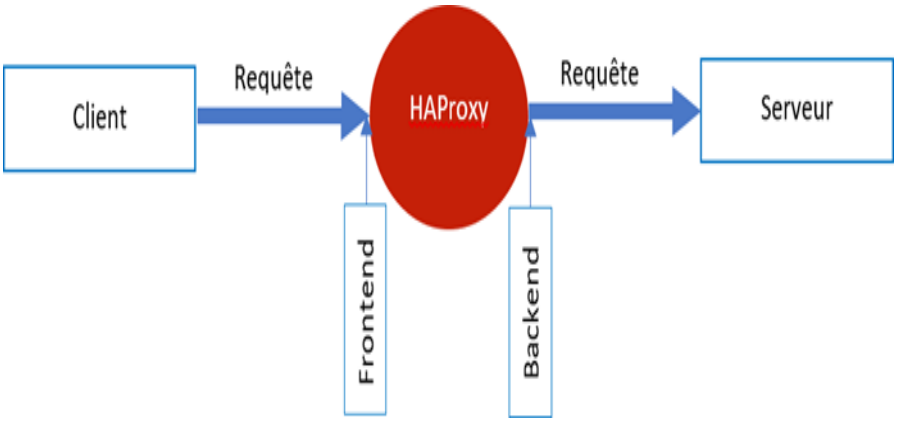


Figure 3 : Fonctionnement de HAproxy.

N'importe quelle machine pourra ainsi tomber en panne sans que l'ensemble ne soit pénalisé.

Ces techniques seront mises en œuvre via deux outils à installer et configurer :

- **Corosync** qui permet de détecter la défaillance d'un poste grâce à un système de communication et de gérer le cluster en lui-même (on aurait pu tout aussi bien utiliser ici un autre outil comme « Heartbeat »).
- **Pacemaker** qui est un gestionnaire de ressources. Il est chargé de créer, démarrer, arrêter et superviser les ressources du cluster c'est-à-dire les services gérés en cluster et donc inclus dans la continuité de services.

6. Comparaison entre HAProxy et LVS :

6.1.1. LVS :

❖ Points forts :

✓ Implantation de LVS sur le noyau, ce qui entraîne un temps de réponse minimal.

✓ Nécessite une faible consommation CPU étant donné qu'il se contente de router les paquets sans examiner leur contenu.

✓ Installé par défaut sur Linux.

❖ Points faibles :

✓ Absence de health check : nécessité d'installer et de configurer Keepalived.

✓ Nécessité de configurer le load balancer comme étant la passerelle par défaut des serveurs backend.

✓ Dans le cas de LVS NAT, le load balancer peut constituer un goulot d'étranglement si le nombre de serveurs backend dépasse 20.

✓ La mise à jour de LVS peut entraîner la mise à jour du kernel de Linux.

6.1.2. HAProxy :

❖ Points forts :

✓ Possibilité de faire un health check au niveau TCP et au niveau applicatif.

✓ Diversité des méthodes pour implémenter la persistance : algorithmes d'ordonnancement (Hachage de l'adresse source par exemple), concaténation de l'ID du serveur avec le cookie d'une application (Le session ID d'une application PHP…), insertion d'un autre cookie, etc.

✓ Possibilité d'utiliser des ACL.

✓ Flexibilité en termes de configuration.

- ✓ Affichage des statistiques sur une interface Web.
- ✓ Fonctionnalité de reverse proxy HTTP.

❖ Points faibles :

- ✓ Consommation CPU assez élevée (relativement à LVS).
- ✓ Nécessite des dépendances pour l'installation.

HAProxy est très flexible et propose un large éventail de fonctionnalités, il permet en outre d'offrir un certain niveau de sécurité grâce au reverse proxying et aux ACL.

Il offre également des outils de validation HTTP (permettant de vérifier si les requêtes sont conformes aux standards HTTP) et une protection au niveau applicatif contre les attaques DDoS. Cet aspect, bien que nous ne l'ayons pas détaillé dans le projet, constitue un réel avantage.

Nous pouvons dire que HAProxy est la solution la plus adaptée.

7. Disponibilité des données :

Même si notre système n'est pas critique et peut supporter un arrêt de service à durée variable, il est généralement inhabituel de se satisfaire de la perte de données. Dans ce cas précis, toutes les techniques utilisées convergent pour garantir l'intégrité des données.

On va s'attacher à décrire les principes fondamentaux qui régissent l'étude et le développement de systèmes critiques Hautement Disponibles (résumé sous l'acronyme HA pour High Availability).

Nous verrons d'abord de façon succincte comment nous pouvons aborder le problème d'un point de vue théorique pour ensuite élaborer une architecture hautement disponible à partir de différents composants logiciels choisis dans une liste non exhaustive. Dans cette partie précise, les briques logicielles que nous utiliserons sont pour la plupart basées sur les logiciels libres : la tentation est donc forte de vouloir mettre en œuvre soi-même, et à peu de frais, des solutions de ce type au sein d'une entreprise. [1]

Il existe **deux types de haute disponibilité de données : les données partagées et les données répliquées**. Les données partagées le sont dans le domaine du réseau. Les données répliquées appartiennent à deux domaines : celui du réseau (réplication serveur à serveur) ou local (réplication disque à disque). Cependant, dans tous ces domaines, un domaine est prédominant.

Voici les différents types de systèmes de fichiers répartis par **domaine** :

7.1. Le RAID :

Le RAID n'est pas une solution 100% viable dans un système de haute disponibilité. Parce que le système devient viable si et seulement si vous êtes sûr que votre serveur ne tombera pas en panne.

Le système Raid n'aidera que pour un secours disque, c'est-à-dire que lorsqu'un disque ne fonctionne plus correctement, un autre prend le relais. Mais au cas où le serveur tombe entièrement, le raid ne pourra plus faire grand-chose. Malgré tout, le Raid reste une solution très utile et à privilégier lorsqu'on le peut. [7]

7.2. Le système de fichier Ext3 :

L'Ext3 est un système de fichiers journalisé. Il est le remplaçant de l'ext2. Ce système de fichiers a évolué principalement à cause des durées souvent trop longues lors d'une vérification du système de fichiers lorsque le serveur n'a pas réussi à démonter proprement les partitions (généralement après un plantage du serveur).

Le grand avantage du Ext3 par rapport aux autres systèmes de fichiers est que l'on passe de l'Ext3 à l'Ext2 et inversement sans problème et sans avoir à jouer avec les différentes partitions pour garder ses données. [8]

7.3. Le système de fichier ReiserFS :

Le ReiserFS est aussi un système de fichiers journalisé. Ce dernier se distingue par le fait qu'il est basé sur une notion d'arbre. De plus, il gagne en performance pour un nombre important de fichiers dans un même répertoire et en espace pour les petits fichiers (sous d'autres filesystems, chaque fichier prend un block au minimum, tandis que le ReiserFS essaye de caser tout dans un seul si le fichier fait moins d'un block). [8]

Ce système de fichiers est efficace mais plus difficilement applicable sur un système déjà existant.

7.4. Le système de fichier InterMezzo :

Le système de fichier InterMezzo est quelque peu différent de ses précédents. InterMezzo permet une réplication par réseau des données.

Il intègre une gestion de déconnexion (si l'un des serveurs de sauvegarde est indisponible, il sera resynchronisé plus tard) et gère l'Ext3, le ReiserFS et le XFS pour l'instant. InterMezzo s'inspire du fonctionnement de Coda.

Très utile dans un système de haute disponibilité, son grand désavantage, c'est qu'il est encore en développement à l'heure actuelle. [8]

7.5. Le NBD :

Network Block Device (NBD) reprend le principe d'InterMezzo et de Coda, dans le sens où il effectue une copie conforme d'un serveur à un autre serveur au moyen du réseau. A la seule différence qu'il n'utilise qu'Ext 2 et NFS nativement. [8]

7.6. Le NFS :

Le NFS (Network File System) procède différemment d'InterMezzo, Coda et autres NBD car il n'effectue pas une réplication de données mais plutôt un partage de données (data shared).

Le système de données ne se trouve pas sur les serveurs de services mais sur un autre serveur dédié. Le gros point noir de NFS est la sécurité : les discussions entre le serveur et son client ne sont pas protégées et les permissions laissent à désirer.

Une solution envisageable consiste soit à utiliser du Tunneling soit à utiliser directement sNFS (Secure Network File System) Cette solution est, malgré tout, recommandé dans certaines solutions de haute disponibilité. [8]

7.7. Le GFS :

Global File System (GFS) est un système de fichiers sous Linux permettant de partager les disques d'un cluster. GFS supporte la journalisation et la récupération de données suite à des défaillances de clients.

Les noeuds de cluster GFS partagent physiquement le même stockage par le biais de la fibre optique ou des périphériques SCSI partagés. Le système de fichiers semble être local sur chaque noeud et GFS synchronise l'accès aux fichiers sur le cluster.

GFS est complètement symétrique ce qui signifie que tous les noeuds sont équivalents et qu'il n'y a pas un serveur susceptible d'être un entonnoir ou un point de panne. GFS

utilise un cache en lecture écriture tout en conservant la sémantique complète du système de fichiers Unix.

Mis à part que GFS s'adresse directement aux personnes ayant les moyens car la fibre d'optique ou les périphériques SCSI ne sont pas bon marché. Hormis ce problème, GFS est un atout indispensable dans une solution de haute disponibilité mais surtout dans le cadre d'un Cluster. [8]

7.8. Le DRBD :

Le DRBD, comme InterMezzo et NBD, effectue un réplica parfait du disque primaire sur un autre disque dur d'un serveur tiers par voie réseau. DRBD est indépendant du type de système de fichiers utilisé sur le serveur.

Vous pouvez donc utiliser n'importe quel système de fichiers.

Tout comme ses congénères, DRBD propose deux types de synchronisation : partielle ou totale. **La synchronisation partielle** n'effectue une mise à jour du disque secondaire que dans les parties non synchronisées (si le serveur a planté par exemple, rien ne sert de refaire une copie du disque primaire).

La synchronisation totale, elle, effectue une copie disque à disque complète, comme si le disque secondaire venait d'être installé.

DRBD pour **Distributed Replicated Block Device** est comparable à un **RAID 1** mais en réseau, c'est à dire que deux disques, partitions ou même un LVM peuvent être répliqué d'un disque à un autre via un réseau Ethernet ou fibre optique.

Cela permet donc d'assurer la **disponibilité** de vos données en cas de crash complet d'une machine. Ce que ne permet pas de faire un RAID classique. [8]

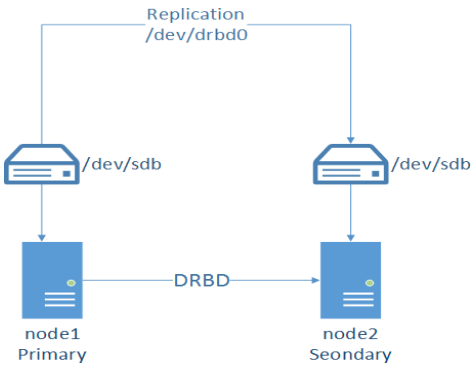


Figure 4 : Schéma DRBD. [8]

7.9. Le SAN/NAS :

SAN (Storage Area Network) et NAS (Network Attached Storage) sont des serveurs dédiés au stockage de données.

Toutefois, SAN et NAS sont différents mais complémentaires. En effet les deux types peuvent être utilisés en même temps pour un service de haute disponibilité. Le NAS se charge du réseau et SAN se charge des serveurs de données.

SAN utilise un protocole SCSI tandis que NAS utilise un protocole IP.

SAN est plus une extension pour serveur alors que NAS est plus dans un optique réseau.

SAN est plus rapide du fait de sa connectique alors que son homologue NAS améliore grandement le modèle de stockage.

Ainsi, ces différentes informations recueillies suivies des contributions des revus documentaires nous amènent à faire une synthèse et choix de la solution pour la mise en place. [8]

7.10. La synchronisation des données :

- ✓ Pour assure la synchronisation des données, il faudra donc envisager une solution pour que le nouveau serveur dispose des données utilisateur.
- ✓ Quelques pistes :
  - Partage NFS (peu recommandé : fiabilité moyenne)
  - Utilisation de DRBD : Distributed Data Block Device (RAID 1 réseau)
  - Baie ou disque SCSI partagé (1 seul accès en écriture à un moment donné)
  - SAN avec switch FC-AL
  - **Synchronisation logicielle au moyen de rsync.**
  - Solution GFS (Global File System) avec un serveur de Stockage. [3]

7.11. La synchronisation avec rsync :

On utilise dans mon projet La synchronisation avec **rsync** :

- ✓ Rsync est une solution open-source très efficace pour effectuer de la synchronisation dedonnées.
- ✓ Rsync permet de mettre à jour des fichiers/répertoires ayant changé sur deux machines mais il ne transfère que les parties de fichiers ayant changé et non la totalité du fichier => ce qui diminue la Bande Passante utilisée
- ✓ Il est très utilisé pour les miroirs de site web et FTP.
- ✓ Il est disponible en rpm pour toutes lesdistributions.
- ✓ Il existe une version Win32 de rsync.

- ✓ Mise en œuvre :
  - Rsync peut être lancé périodiquement avec une commande cron
  - Le plus simple est d'utiliser un tunnel **ssh** pour le transfert des données (on créera une clé publique sur le client que l'on exportera sur le serveur maître).
  - Exemple : rsync -az -e ssh master:/home/ /home
  - Remarque : pour Samba, il faudra également synchroniser les fichiers /etc/passwd, /etc/groupet smbpasswd

Remarque :

- Il sera également nécessaire de prévoir des scripts pour synchroniser le maître depuis l'esclave.
- En effet quand le maître sera à nouveau en ligne, il devra récupérer les nouveaux fichiers depuis l'esclave.

	Rsync	Serveur de fichiers
Avantages	• Envoie juste les parties des fichiers ayant changé, non tout le fichier. • Peut compresser les données pour réduire encore le trafic.	• Synchronisation en temps réel (ce qui est écrit sera retrouvé par l'autre machine en cas de crash). • Peu gourmand en ressources processeur.
Inconvénients	• Sauvegarde programmée (par exemple toutes les 5 minutes). Les fichiers écrits entre temps sont perdus en cas de crash. • Programme très gourmand en ressources processeur.	• Nécessite une machine supplémentaire ou un équipement de stockage réseau.

Tableau 2 : Comparaison entre La synchronisation avec rsync et la synchronisation avec serveur de fichier :

8. Comparaison entre quelques logicielles existant de Haute Disponibilité :

Quelque produit étudié est récapitulés dans le tableau suivant :

Produit	Version
Linux	2.2.15
Mon	0.38.18
Heeartbeat	0.4.7b
DRBD	0.5.5
ReiserFS	3.5.21-2.2.15
GFS	Version du 23/11/2019
LVS	0.9.13-2.2.15

Tableau 3 : Quelque produit de Haute Disponibilité.

Mon :

Mon est un outil de monitoring généraliste. Il permet de surveiller des ressources (services réseau, ressources locales) et de déclencher des actions selon leur état (lancement de scripts, notification par E-Mail, pager, envoi de *traps*). [9]

HACMP :

HACMP (High-Availability Cluster MultiProcessing) est une technique de haute disponibilité utilisée par IBM, à base de Reliable Scalable Cluster Technology. En 2013, la version 7.1 est disponible pour AIX. La version 5.4 est disponible pour Linux sous licence IBM (le type exact de licence est IPLA). [9]

Trucluster :

Est une solution de clustering haute disponibilité à source fermée pour le système d' exploitation Tru64 UNIX. Développé à l'origine par Digital Equipment Corporation (DEC), il a ensuite été transféré à Compaq en 1998 lors de l'acquisition de Digital par la société, qui a ensuite fusionné avec Hewlett-Packard (HP). [9]

Cette partie présente les fonctionnalités identifiées dans le cadre de la haute disponibilité. Certaines seront décomposées en sous-fonctions.

Ces fonctionnalités sont les suivantes :

- ✓ **Mise en cluster** :

Ce thème recouvre l'ensemble des fonctions permettant la reprise d'un service assurée par un noeud d'un cluster par un autre noeud en cas de défaillance matérielle ou logicielle.

Les sous-fonctions sont détaillées ci-dessous :

- **Mécanisme de reprise** : tous les mécanismes permettant de reprendre un service qui a été interrompu par une panne. Cela inclut le lancement d'applications et la reprise de données et d'adresses réseau. [9]

- **Détection de pannes :** détection de problèmes matériels ou logiciels pouvant déclencher une action correctrice comme par exemple le basculement d'un service sur un noeud qui n'est pas touché par le problème. [9]

✓ **Disponibilité des données :** Cette fonction garantit la haute disponibilité des données stockées sur disque, elle inclut les mécanismes de redondance des données et la reprise. Les sous-fonctions identifiées sont les suivantes :

- **Support du RAID :** indique le support de solutions matérielles existantes RAID, ou la fourniture d'une fonctionnalité équivalente purement logicielle. [9]

- **Disques partagés entre plusieurs noeuds :** support des solutions matérielles de disques partagés où plusieurs noeuds sont directement reliés aux disques concernés. [9]

- **Volumes partagés :** présence de mécanismes permettant d'utiliser des périphériques distants, ou de répliquer "au fil de l'eau" un périphérique local sur un noeud distant. [9]

- **Haute disponibilité des systèmes de fichiers distants :** inclut tous les mécanismes permettant la haute disponibilité d'un système de fichiers distant (tel que NFS) dans le cas où le serveur fournissant ce système de fichiers tombe en panne. [9]

- **Reprise après panne :** inclut le support de mécanismes permettant de reprendre un système de fichiers après un "crash" sans perte de données. [9]

✓ **Redondance matérielle locale :** Ce thème consiste à évaluer le support, par les produits étudiés, des solutions matérielles de haute disponibilité telles que celles trouvées sur les serveurs Unix haut de gamme. Les solutions matérielles incluent :

- Périphériques redondants (stockage, réseau, alimentations, CPUs, utilisation de crossbar pour rendre le bus redondant).

- Changement "à chaud" de périphériques.

- Utilisation de watchdogs (périphériques permettant de déclencher un redémarrage matériel en cas de "plantage" du système d'exploitation).

- Gestion des onduleurs.

- Support d'outils assurant l'arrêt ou le redémarrage d'un noeud à distance depuis une application tournant sur un noeud distant. [9]

✓ **Equilibrage de charge :**

Cette fonction permet de répartir des requêtes adressées à un service, entre plusieurs noeuds physiques fournissant ce service. Cette fonction est généralement utilisée pour des services réseaux tels que serveurs Web, messagerie, DNS.

Elle entre dans le cadre de la haute disponibilité car un produit d'équilibrage de charge ne redirigera pas les requêtes vers un serveur en panne. L'intérêt d'une telle solution est que la reprise est immédiate (il n'y a pas d'application à démarrer sur un noeud de backup lorsqu'une panne est détectée). [9]

✓ **Redondance multi-sites :** Cette fonction assure la haute disponibilité en dupliquant une application et ses données sur des sites distants reliés par un WAN, et permet, en cas de panne sur un site, la reprise du service fourni par l'application par un autre site.

Le tableau ci-dessous présente, pour chacune des fonctions, la liste des produits qui assurent cette fonction :

Fonctions	Logiciels libre fournissant cette fonction
Mise en cluster	
Mécanismes de reprise	Heartbeat
Détection de panne	Heartbeat, Mon
Disponibilité des données	
Support du RAID	Linux (pour RAID matériel et logiciel)
Disques partagés	GFS
Volumes partagés	DRBD
Haute disponibilité des systèmes de fichiers	ReiserFS
Equilibrage de charge	
Equilibrage de charge	LVS
Redondance matérielle locale	
Redondance matérielle locale	Linux
Redondance multi-site	
Redondance multi-site	Aucun

Tableau 4 : Couverture fonctionnelle des produits. [10]

Le tableau ci-dessous présente la Comparaison entre quelques logicielles existant de Haute Disponibilité :

Critères	Evaluation			Logiciel libre	
	HACMP	TruCluster	Log. Libre	Produits	Remarques
Mise en cluster					
Mécanismes de reprise	***	***	*	Hearbeat	Heartbeat fourni une solution "légère" de *clustering* pour deux serveurs. Ce produit est à compléter par un outil de détection de pannes logicielles tel que Mon.
Modes de "clusterisation" supportés	+	+	-		Contrairement aux solutions commerciales, Heartbeat ne supporte qu'un seul mode de reprise En particulier, pas de failover en cascade, d'instances multiples de service? . Heartbeat impose un basculement lorsque le noeud principal est disponible après une panne.
Nombre de noeuds supportés	32	8	2		La documentation sur Heartbeat indique que la limitation à 2 noeuds devrait être bientôt supprimée (l'architecture du produit est prévue pour un plus grand nombre de noeuds).
Sélection dynamique du noeud de backup	+	-	-		Cette fonction n'est pas implémentée dans Hearbeat et TruCluster.
Compatibilité avec les produits d'équilibrage de charge	+	+	-		Heartbeat ne supporte pas d'instances multiples de service. Il n'est donc pas possible d'utiliser Heartbeat pour gérer les serveurs dans le cadre d'une solution d'équilibrage de charge à l'aide de ce produit.

Actions correctrices possibles	+	+	+		Actions possibles : • Reprise d'adresse IP ; • Reprise d'un service (via script de démarrage) ; • Action quelconque (script à fournir). En cas de panne d'une interface réseau le produit ne sait pas basculer sur une autre interface (sauf à développer cette fonction). Les trois produits sont équivalents.
Détection de pannes	***	***	***	Heartbeat, Mon	Les logiciels libres étudiés permettent de surveiller la plupart des ressources logicielles et matérielles.
Détection pannes logicielles	+	+	+		Mon permet de surveiller un grand nombre de services réseau (HTTP, LDAP?).
Détection pannes matérielles	+	+	+		Les produits permettent de détecter des pannes matérielle avec des mécanismes de "ping" (pour Mon) ou de dialogue entre agents présents sur les noeuds. Les trois produits sont équivalents. On peut noter que Heartbeat dialogue par plusieurs méthodes simultanément (réseau et port série), ce qui lui permet de distinguer les pannes réseau et pannes complètes d'un noeud. Le produit lm-sensors (non étudié dans ce rapport) permet de contrôler les sondes de température, ventilateurs, alimentation, présent dans les matériels PC standards.

Détection de problèmes de ressources	+	-	+	La seule ressource qu'on sait surveiller (avec Mon) est l'espace disque, mais on peut en ajouter en écrivant ses propres scripts. Au niveau des solutions commerciales, cette solution est implémentée dans le produit d'IBM mais pas celui de Digital.
Interfaces (API)	+	+	+	**Heartbeat :** Il est possible de déclencher le basculement depuis une application par l'intermédiaire d'outils en ligne de commande. L'API de consultation de l'état du cluster est en cours de développement et présente encore de gros problèmes de stabilité. **Mon :** Mon est livré avec un outil d'administration en ligne de commande que l'ont peut exécuter sur le noeud du serveur Mon ou à distance. Les fonctions de l'outil sont décrites Le produit est livré avec des exemples de scripts utilisant cet outil. Les APIs des produits commerciaux sont plus homogènes.
Extensibilité	+	+	+	**Heartbeat :** On peut ajouter des actions à déclencher lors du basculement. **Mon :** On peut ajouter des scripts de surveillance de nouvelles resources ou des scripts à déclencher en cas d'alerte. Les trois produits sont équivalents.
Réglages possibles			+	Tous les intervalles de surveillance, le nombre d'échecs avant alerte ou basculement sont paramétrables. Dans le cas de Mon on peut définir des dépendances entre les resources partagées pour limiter le nombre d'alarmes.

Disponibilité des données					
Support RAID	**	**	**	Linux	Linux fournit une fonction de RAID logiciel très complète, mais supporte très peu de solutions matérielles.
RAID matériel	+	+	-		Un seul produit (contrôleur) supporté par Linux contrairement aux autres solutions..
RAID logiciel	-	-	+		Très complet pour Linux. Pas supporté par HACMP. Support limité par Digital (pas de Raid 5).
Disques partagés	**	***	**	GFS	Solutions commerciales : • Compaq supporte CFS (Cluster File System) ; • IBM supporte la même fonctionnalité mais de façon non conforme à la norme POSIX. • GFS supporte peu de matériels.
Produits supportés	+	+	-		GFS impose des limitations au matériel : le disque ou la baie doivent supporter la spécification Dlock qui est peu répandue, et les constructeurs qui l'ont implémenté ne la supportent pas. Support SCSI et Fibre Channel uniquement (pas de SSA).
Accès concurrent possible	-	+	+		Note : GFS supporte la journalisation dans le cas d'accès en écriture multiples, contrairement à la solution d'IBM.
Nombre de noeuds supportés	8		16+		La mise en place de GFS la plus importante a utilisé 16 noeuds accédant de façon concurrente à des disques partagés.
Volumes partagés	*	*	***	DRBD	Cette fonctionnalité n'est pas implémentée dans les solutions commerciales où on utilise plutôt des disques partagés.
Mécanisme de réplication de volumes / utilisation de volumes distants	n/a	n/a	+		Le seul mode de fonctionnement supporté par DRBD est la réplication à la volée.
Accès concurrent possible	n/a	n/a	-		Pas d'accès concurrent possible
Mécanisme de resynchronisation après déconnexion	n/a	n/a	+		DRBD a 2 mécanismes : • resynchronisation partielle déclenchée automatiquement après une déconnexion temporaire (coupure réseau ou panne du secondaire) • resynchronisation totale (recopie de l'ensemble du volume) : Son déclenchement se fait manuellement par un outil en ligne de commande. Les deux mécanismes fonctionnent en tâche de fond.

Nombre de noeuds supportés	n/a	n/a	2		La *roadmap* du produit indique que la limitation a 2 noeuds sera supprimée.
Haute disponibilité des systèmes de fichiers	***	***	***	ReiserFS	ReiserFs : Journalisation des méta-données uniquement (c'est aussi le cas des solutions commerciales), ce qui correspond à une utilisation courante.
Systèmes de fichiers journalisés	+	+	+	ReiserFS	Pour les solutions commerciales, la journalisation est une fonction standard des systèmes d'exploitations sur lesquelles elles fonctionnent. Note : plusieurs autres systèmes de fichiers journalisés sont en cours de développement (XFS, JFS, ext3, LinLogFS?)

Haute disponibilité des systèmes de fichiers	+		-	***Aucun***	IBM fournit des fonctions de récupération de l'état d'un système de fichiers (verrous?) après un crash. Aucun des produits libres ne fournit de solution.
Equilibrage de charge					
Equilibrage de charge	***		***	LVS	La solution commerciale considérée pour IBM est : Network Dispatcher pour IBM/HACMP
Haute disponibilité du mécanisme d'équilibrage de charge	+		+		Ça n'est pas une fonction du produit de base : il faut utiliser une des solutions Piranha ou Ultramonkey, ou cela doit être réalisé par ailleurs.
Critères de "dispatching" supportés	+		-		Critères : • round robin pondéré ou pas, • choix du serveur avec le moins de connexions actives, pondéré ou pas. LVS ne supporte pas de critère dépendant de l'état des serveurs (charge, utilisation CPU ..) contrairement à la solution d'IBM.
Détection des noeuds en panne	+		+		Les solutions Piranha et Ultramonkey apportent cette fonction.
Mécanismes supportés	-		+		LVM est la solution qui supporte le plus de mécanismes de routage de paquets vers les serveurs. IBM ne supporte que le routage direct.
Redondance matérielle locale					
Redondance matérielle locale	***		**	Linux	Rien n'est prévu parmi les logiciels libres, à part la gestion des watchdogs et des onduleurs.
Support des matériels hautement disponibles	+		-	***Aucun***	Pas de solution logiciel libre.
Gestion des onduleurs	+		+	Cf. remarque	Il existe de nombreux produits dans le logiciel libre permettant de gérer les onduleurs et de déclencher des actions selon l'état de l'onduleur. Ces produits n'ont pas été étudiés dans ce document.

Support des watchdogs			+	Linux	Supporté par Linux, mais le nombre de solutions matérielles supportées est très faible (3 produits).
Support arrêt+redémarrage matériel	+		-	***Aucun***	Disponible sur HACMP mais pas de solution logiciel libre.
Redondance multi-site					
Redondance multi-site	***	*	*	***Aucun***	Aucun logiciel libre ne fournit cette fonctionnalité (parmi les solutions commerciales, seul IBM fournit une fonction de reprise multi-site sans limite de distance via un WAN).
Mécanismes de synchronisation de données entre sites	+	n/a	n/a		
Mécanismes de reprise de multi-sites	+	n/a	n/a		
Critères généraux					
Notoriété	***	***	*	tous	Les produits logiciels libres étudiés (à part le système Linux) présentent peu de références à l'heure actuelle. On peut citer l'utilisation de ReiserFS pour le serveur sourceforge.net (850Go de fichiers en ligne dont la moitié est sous ReiserFS).
Qualité technique	***	***	**		Variable selon les logiciels libres utilisés.
Robustesse	+	+	+		Variable selon les produits. Certains produit nécessitent des corrections pour pouvoir fonctionner correctement. Par exemple dans certaines circonstances, l'utilisation conjointe de DRBD et Heartbeat conduit à des situations ou les deux noeuds essaient d'être primaire en même temps, ce qui conduit au blocage du produit DRBD.
Sécurité	+	+	-		
Administration et exploitation	***		*		Admninstration très "Bas niveau" pour les logiciels libres.
Configuration à chaud possible	+		-		Pas de configuration sans redémarrage des produits, à part : • Retrait temporaire d'un serveur d'un cluster Heartbeat possible ; • Arrêt de la surveillance d'une ressource par Mon.
Configuration centralisée	+		-		Aucune pour les logiciels libres.

API d'administration	+		-	Logiciels libres : en général, administration "à l'ancienne" : fichier de configuration et redémarrage du produit pour prendre en compte les modifications. Seul Mon fournit une vraie API. IBM fournit une API C/C++.
Centralisation de la configuration des systèmes d'exploitation des différents noeuds	+		-	Utilisation des mécanismes standard Unix pour les logiciels libres. Les outils d'administration d'IBM permettent d'administrer les systèmes d'exploitation des différents noeuds d'un cluster de manière centralisée (gestion des utilisateurs, nom des serveurs ?)
Console d'exploitation centralisée	+		-	IBM fournit une console graphique présentant l'état des différents noeuds d'où on peut déclencher des actions telles que le basculement d'un service, le démarrage ou l'arrêt d'un noeud ? Rien pour les logiciels libres.
Mécanismes d'alertes	+		-	Seul Mon est capable de remonter ses propres alertes. N'existe pas sur les autres logiciels libres..
Maintien en conditions opérationnelles	***		**	Documentation "dense" et très technique et * support » par mailing-list pour les logiciels libres..
Documentation	+		+	Documentation type Man UNIX ou HOWTO, en général assez concise et pas du tout didactique, mais assez complète. Les produits sont fournis avec des exemples de fichiers de configuration. Le produit LVS se démarque des autres en fournissant une documentation en ligne de bonne qualité contenant plusieurs exemples de configuration. Bonne documentation pour HACMP.
Support	+		-	Pas de support officiel pour le logiciel libre, mais il y a une mailing-list sur la haute disponibilité (archive avec outil de recherche et modalités d'inscription sur www.linux-ha.org) à laquelle les développeurs des produits participent.
Prise en main	+		-	Les produits sont en général simples à installer (à part ReiserFS qui nécessite un patch noyau et dont la mise en place comme système de fichier d'une distribution existante peut être délicat) et à utiliser. Mais une intégration est nécessaire pour les logiciels libres.

Critères	Notation			Synthèse logiciel libre	
	HACMP	TruCluster	Log. libre	Produits	Remarques
Mise en cluster					
Mécanismes de reprise	***	***	*	Heartbeat	Heartbeat fournit une solution "légère" de clustering pour deux serveurs.
Détection de panne	***	***	***	Heartbeat, Mon	Les logiciels libres étudiés permettent de surveiller la plupart des ressources logicielles et matérielles.
Disponibilité des données					
Support RAID	**	**	**	Linux	RAID logiciel plus complet que les solutions commerciales. RAID matériel supporte peu de contrôleurs.
Disques partagés	**	***	**	GFS	Produit complet mais avec des contraintes importantes sur le matériel (support d'une extension à la norme SCSI).
Volumes partagés	*	*	***	DRBD	DRBD fournit une fonction de réplication de disque au fil de l'eau et n'a pas d'équivalents dans les solutions commerciales.
Haute disponibilité des systèmes de fichiers	***	***	***	ReiserFS	ReiserFS est déjà utilisé en production sur plusieurs gros serveurs.
Equilibrage de charge					
Equilibrage de charge	***		***	LVS	LVS est disponible sous forme de solutions complètes,les outils administration viennent de sortir.
Redondance matérielle locale					
Redondance matérielle locale	***		**	Linux	Linux ne supporte pas la plupart des matériels redondants, mais permet de gérer les onduleurs et les watchdogs.
Redondance multi-site					
Redondance multi-site	***	*	*	***Aucun***	Aucun logiciel libre ne fournit cette fonctionnalité (parmi les solutions commerciales, seul IBM fournit une fonction de reprise multi-site sans limite de distance via un WAN).
Critères généraux					
Notoriété	***	***	*	Tous	Très peu de références en production actuellement (à part Linux).
Qualité technique	***	***	**		Variable selon les produits.
Administration	***		*		Adrminstration très "Bas niveau" pour les logiciels libres à l'exception des deux solutions intégrant LVS.
Maintien en conditions opérationnelles	***		**		Documentation "dense" et très technique et support par mailing-list.

Tableau 5 : Comparaison entre quelques logicielles existant de Haute Disponibilité. [10]

On constate que dans l'environnement des logiciels libres on trouve encore peu de produits fournissant une solution globale de haute disponibilité, et que pour mettre en place une telle solution il faut utiliser plusieurs "petits" produits apportant chacun une fonction de la haute disponibilité (clustering, réplication de disques, RAID, …). Les produits doivent ensuite être intégrés.

Conclusion :

Dans ce chapitre, nous avons défini le champ de notre étude. Nous avons aussi présenté les différents approches et logicielles de la haute disponibilité, suivi d’une étude critique afin de réaliser notre architecture.

Dans le chapitre suivant, nous présenterons l’architecture technique de notre solution.

Chapitre 2 : Etude de l'existant et conception de la solution

Introduction :

Ce chapitre décrit l'étude de l'existant et la conception de notre solution (nous proposons et présenterons l'architecture technique de notre solution).

1. Ancien architecture de la plateforme de PROGRES de MESRS :

La plate-forme se décompose en deux parties. La première partie représente le site central de production.

La seconde partie découle naturellement de la première et concerne la plate-forme de récupération en cas de désastre (Disaster Recovery (DR)) avec une réplication asynchrone.

La première partie se compose d'un mixte de serveurs physiques et de serveurs virtuels créaient au niveau MESRS.

La deuxième partie est strictement physique, au niveau du CERIST.

La configuration globale de l'ancienne plate-forme PROGRES est comme suit :

Les serveurs d'application (App_Server) sont connectés directement aux routeurs et reliés directement avec un serveur des bases de données (BDD_Master_Server) et un serveur de stockage (NFS_Server).

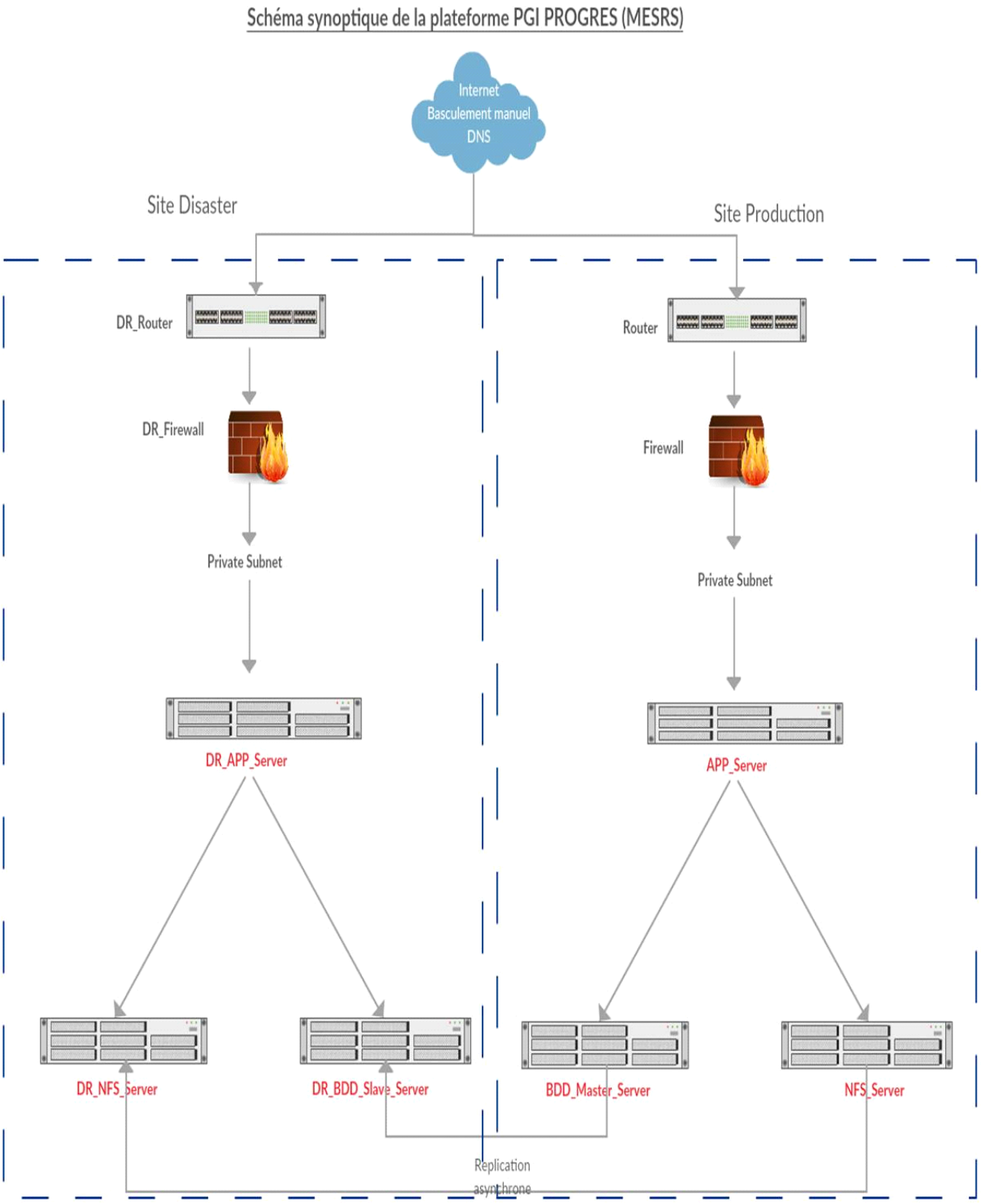


Figure 5 : Ancien schéma synoptique de la plateforme PGI PROGRES (MESRS).

Selon le problématique que l'on a déjà abordé en l'introduction générale et le chapitre précédent on aurait dû revoir toute l'architecture des systèmes et l'infrastructure de services afin de trouver la meilleure solution qui répond et répondra aux demandes changeantes des services actuelles et futures et d'assurer les exigences légales, contractuelles et les exigences des niveaux de service.

2. Choix de la solution :

Mon travail a consisté à revoir l'architecture et à améliorer les serveurs de la plateforme et pour cela j'ai ajouté deux serveurs **HAProxy** et modifié les chemins des autres serveurs.

❖ **Serveurs de load balancing HAProxy :**

- ✓ HAProxy est réputé pour être stable, très fiable, avec de bonnes performances grâce à sa maturité (17 ans d'existence).
- ✓ L'intérêt d'utiliser HAProxy, plutôt qu'un des nombreux autres reverse proxy (Nginx, Squid, LVS …) est qu'il apporte des fonctionnalités très avancées, comme **le filtrage niveau 7 (OSI)**.

Remarque :

Le niveau **« applicatif »** ou **niveau 7 (OSI)** ou **« répartition avec affinité de serveur »** : On analyse ici le contenu de chaque requête pour décider de la redirection. En pratique, deux choses sont recherchées et analysées :

- les cookies, qui figurent dans l'entête HTTP ;
- L'URI, c'est-à-dire l'URL et l'ensemble de ses paramètres.

Ce niveau est parfois rendu nécessaire par certaines applications qui exigent que les requêtes d'un même utilisateur soient adressées à un même serveur.

Cette technologie de répartition induit bien évidemment des délais supplémentaires car chaque requête HTTP doit être analysée.

- ✓ Ainsi HAProxy est très flexible et propose un large éventail de fonctionnalités, il permet en outre d'offrir un certain niveau de sécurité grâce au reverse proxying et aux ACL, Il offre également une protection au niveau applicatif contre les attaques DDoS. [6]

- ✓ La disponibilité à 100% n'existe pas et il ne faut pas viser les 100% car plus on va vouloir rajouter de serveurs, plus le risque va augmenter. Grâce aux communautés et entreprises, un bon nombre de solutions de HA voient le jour et il y en a pour tous les besoins, et **HAproxy est la solution la plus adaptée et la meilleure solution pour tous ces déséquilibres.**
- ✓ HAproxy est Peu cher avec Temps de basculement court.

❖ **Serveur d'application (Apache Tomcat) :**

- ✓ Compatible avec les applications JAVA.
- ✓ Meilleur pour les applications simples et flexibles.
- ✓ Comporte un serveur HTTP.

❖ **Serveur de base de données (Percona Server for MySQL):**

- ✓ Le seul SGBD qui fait le clustering active/active et simple d'utilisation.

Remarque :

Le clustering : est une méthode permettant d'accélérer l'exécution d'un programme informatique en divisant celui-ci en multiples segments exécutés simultanément sur différentes machines.

❖ **Serveur de stockage (GlusterFS) :**

- ✓ Le seul qui fait la réplication et extension à la demande.
- ✓ Pour la réplication bidirectionnelle, il existe GlusterFS , (version open source gratuite).

Remarque :

La réplication a pour but de sécuriser la donnée en la synchronisant entre plusieurs serveurs.

Cette synchronisation peut être **unidirectionnelle ou incrémentale**, c'est-à-dire que la donnée est écrite et accessible mais ne peut être modifiée. Souvent utilisée pour faire de la sauvegarde, cette technique permet d'avoir plusieurs versions d'un même fichier. La seconde possibilité est d'avoir une synchronisation **bidirectionnelle**, ce qui signifie que la donnée sera

clonée entre deux ou plusieurs serveurs mais également accessible et modifiable en temps réel.

Pour éviter que deux clients écrivent en même temps sur le même fichier, des systèmes de verrous et de temporisations peuvent être mise en place.

Un service est installé sur les deux serveurs et synchronise leurs données. Dans le cas de la sauvegarde, il y a un serveur primaire et un secondaire. Le client aura seulement accès au serveur primaire. Dans le cas de **la réplication bidirectionnelle**, le service couplé à un répartiteur de charge, permettra au client d'accéder au site internet sans se douter qu'il est sur un des deux serveurs.

❖ **Serveur de cache (Varnish) :**

- ✓ Décharger de cache HTTP et servir plus rapidement les requêtes.

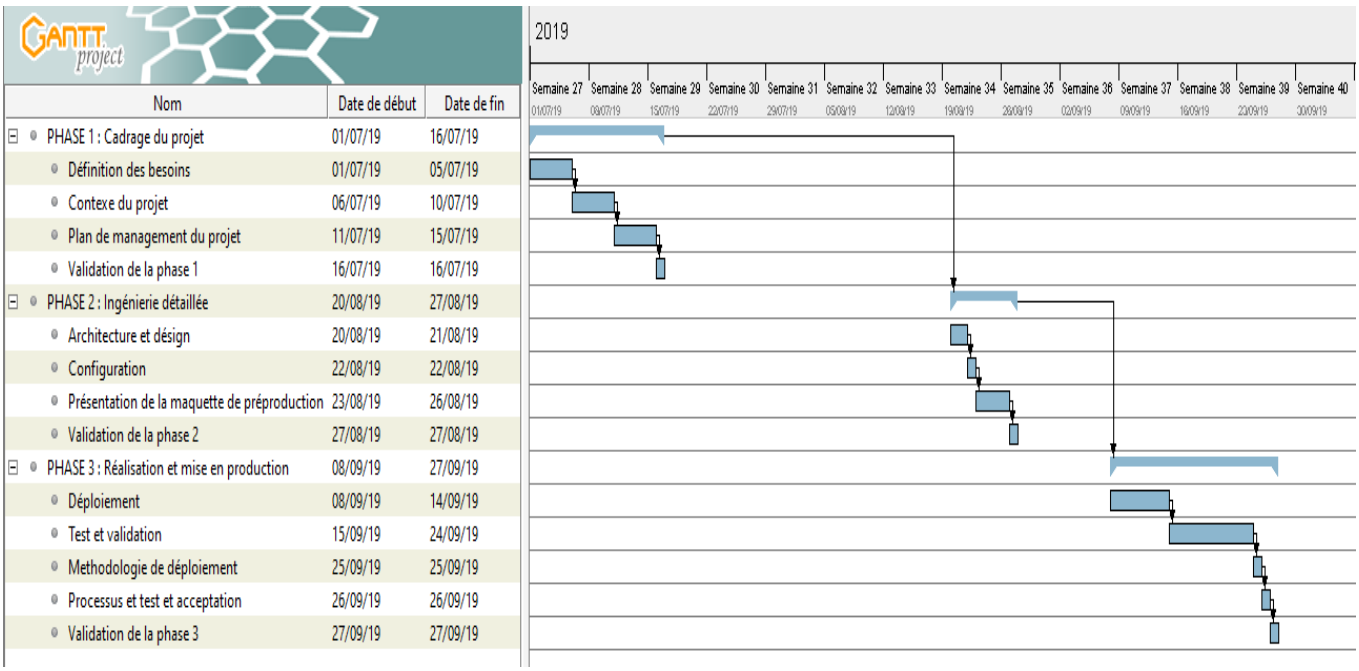
3. Travail réalisé :

Le but de ce projet est d'assurer la disponibilité et la continuité du service d'inscription pour le master et le doctorat du Ministère de l'Enseignement Supérieur et de la Recherche Scientifique (MESRS).

On fait **le diagramme de Gantt**, cet outil répond à deux objectifs : planifier de façon idéal ainsi que communiquer sur le planning établi et les choix qu'il impose.

Le diagramme permet :

- ✓ de déterminer les dates de réalisation d'un projet.
- ✓ d'identifier les marges existantes sur certaines tâches.
- ✓ de visualiser d'un seul coup d'œil le retard ou l'avancement des travaux.



Nom	Date de début	Date de fin
PHASE 1 : Cadrage du projet	01/07/19	16/07/19
Définition des besoins	01/07/19	05/07/19
Contexe du projet	06/07/19	10/07/19
Plan de management du projet	11/07/19	15/07/19
Validation de la phase 1	16/07/19	16/07/19
PHASE 2 : Ingénierie détaillée	20/08/19	27/08/19
Architecture et désign	20/08/19	21/08/19
Configuration	22/08/19	22/08/19
Présentation de la maquette de préproduction	23/08/19	26/08/19
Validation de la phase 2	27/08/19	27/08/19
PHASE 3 : Réalisation et mise en production	08/09/19	27/09/19
Déploiement	08/09/19	14/09/19
Test et validation	15/09/19	24/09/19
Methodologie de déploiement	25/09/19	25/09/19
Processus et test et acceptation	26/09/19	26/09/19
Validation de la phase 3	27/09/19	27/09/19

Figure 6 : Diagramme de Gantt.

3.1. Nouvelle architecture de la plateforme de PROGRES de MESRS :

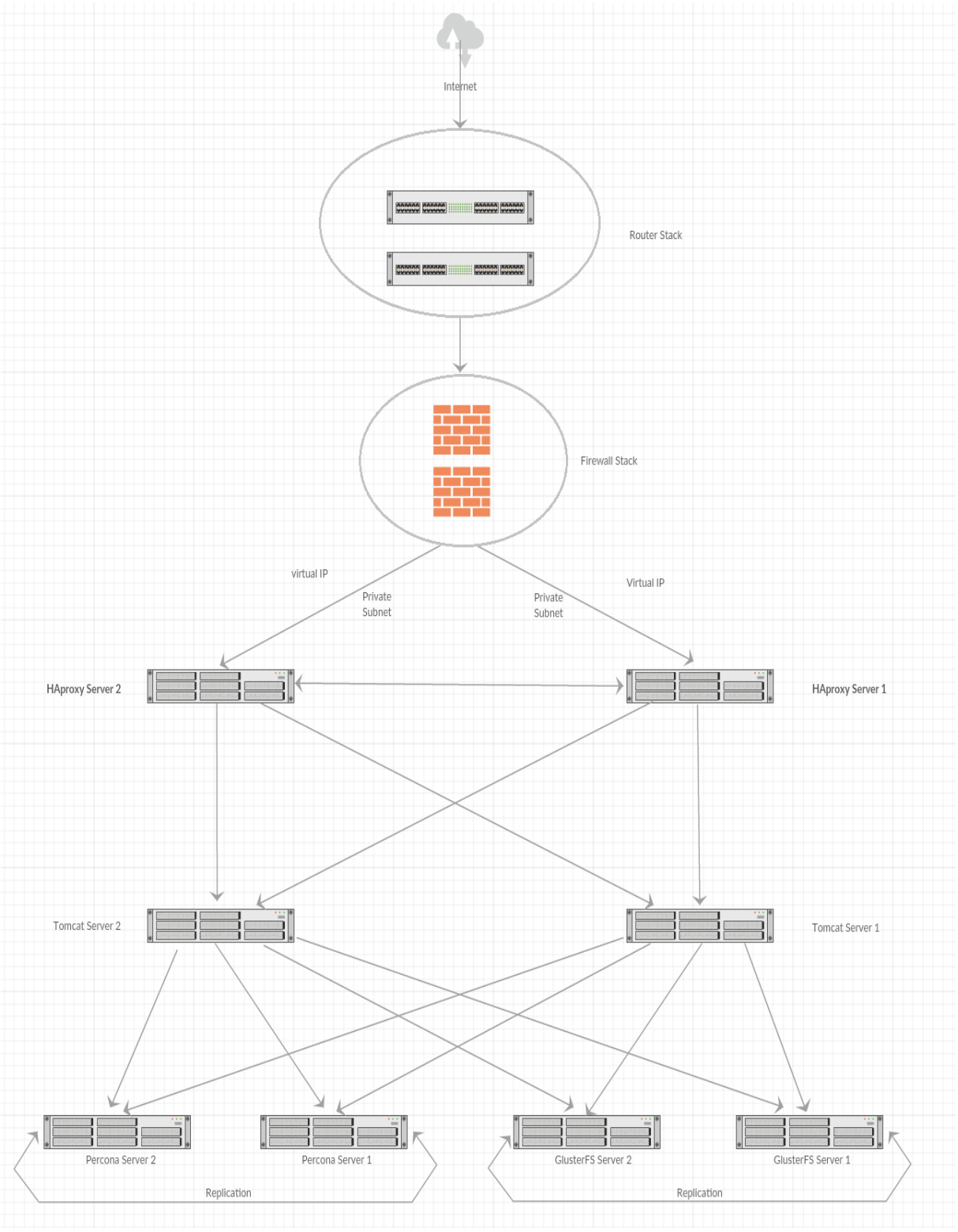


Figure 7 : Nouveau schéma synoptique de la plateforme PGI PROGRES (MESRS).

La configuration globale et le processus de flux de données au niveau de la nouvelle plate-forme PROGRES est comme suit :

Il se constitue de 10 serveurs:

Deux serveurs de load balancing :

Ces serveurs sont utilisé pour faire l'aiguillage entre l'utilisateur du PGI et les serveurs d'applications grasse à la solution **« HAPROXY »**, le tous en mode balancement qui assure la disponibilité de l'application en cas de problème avec l'un deux.

Chaque données ou requête qui arrive au niveau du site maître passera en premier lien traité par l'un des deux serveurs de load balancing « HAPROXY » qui fera à son tour l'aiguillage de l'information vers les serveurs d'applications « TOMCAT » pour y être traité et qui marchent en actif/actif.

- ✓ **Pour éviter tous arrêt du service** et avec lui arrêt de l'application PROGRES les deux serveurs sont installé en cluster avec la solution **« corosync »**, qui assure en permanence l'état du service d'aiguillage vers les serveurs d'applications , donc si par malheur un serveur tombe en panne, le cluster démarre automatiquement l'aiguillage sur le deuxième serveur et vice versa.

Comme ça le service sera toujours opérationnel en permanence.

Deux serveurs d'application« TOMCAT » :

Chaque serveur englobe plusieurs instances du serveur d'application « TOMCAT », ce qui augmente le nombre de serveur d'application qui tournent simultanément à plusieurs instances. Tous ces serveurs travail sur la même base de données (serveur MAITRE) et récupèrent les fichiers du même serveur GlusterFS. Ce qui augmente considérablement le temps de réponse de l'application et diminue le nombre d'erreur liés aux crashs.

- ✓ **Récupération des serveurs d'application :** plusieurs script de démarrage et d'arrêt des serveurs d'applications sont mise au point pour intervenir directement après n'importe quelle anomalie

Deux serveurs de stockage GlusterFS :

Marchent maitre/esclave, si le serveur maitre tombe, le serveur esclave devient maitre et une fois le premier serveur opérationnel il redevient esclave. La sauvegarde et la réplication

des données se fait automatiquement et en temps réel avec **une synchronisation bidirectionnelle,** ce qui signifie que la donnée sera clonée entre deux ou plusieurs serveurs mais également accessible et modifiable en temps réel c.à.d le service couplé à un répartiteur de charge, permettra au client d'accéder au site internet sans se douter qu'il est sur un des deux serveurs.

Et avec une synchronisation avec **rsync** permet de mettre à jour des fichiers/répertoires ayant changé sur deux machines mais il ne transfère que les parties de fichiers ayant changé et non la totalité du fichier (il est très utilisé pour les miroirs de site web).

Deux serveusr de base de données (Percona Server for MySQL) :

Marchent en actif/actif et la réplication des données se fait en temps réel avec la méthode du clustering.

- ✓ Une synchronisation en continu avec le serveur NFS du **CERIST** pour dupliquer les données **en cas de perte**. Si le site maître et hors ligne, le serveur de secours du CERIST fera la relève.

Deux serveurs de caches Varnish :

Ils sont **installés** sur les machines des serveurs HAProxy pour décharger de cache HTTP et servir plus rapidement les requêtes.

Les adresses IPs des serveurs utilisés sont comme suit :

HAProxy 1: **192.168.112.176**
HAProxy 2 : **192.168.112.178**
Tomcat 1 : **192.168.112.177**
Tomcat 2 : **192.168.112.174**
Percona 1 : **192.168.112.175**
Percona 2 : **192.168.112.179**
GlusterFS1 : **192.168.112.185**
GlusterFS 2 : **192.168.112.186**

Deux serveurs de caches **Varnish** installés sur les machines des serveurs **HAProxy 1** et **HAProxy 2**.

Conclusion :

Ce chapitre était consacré à la conception de la plateforme proposée, et nous avons spécifié les fonctionnalités de cette plateforme.

En outre, ce chapitre a décrit la solution conceptuelle qui répond à nos besoins.

Dans le prochain chapitre nous présenterons l’ensemble des outils utilisés (Matériels et Logiciels) ainsi que les processus de la configuration implémentée de la plateforme.

Introduction :

Dans ce chapitre, nous présenterons l'ensemble des outils utilisés (Matériels et Logiciels) ainsi que les processus de la configuration implémentée de la plateforme.

1. Outils utilisés:

1.1. Outils matériels :

Lap top: ASUS ROG GL703V i7 7700HQ 16/1050 4GB

- ✓ INTEL I7 7700HQ.
- ✓ NVIDIA GTX 1050 4GB.
- ✓ 16GB DDR4 2400HZ RAM (POSSIBILITÉ D'AUGMENTATION).
- ✓ 128GB NVME PCIE SSD + 1TB HDD.
- ✓ CLAVIER MULTICOLORE.
- ✓ ECRAN 17" 120HZ.

1.2. Outils logiciels :

1.2.1. VMware Workstation Pro :

VMware Workstation Pro est un outil de virtualisation de poste de travail créé par la société VMware, il peut être utilisé pour mettre en place un environnement de test pour développer de nouveaux logiciels, ou pour tester l'architecture complexe d'un système d'exploitation tels que Microsoft Windows, Linux, NetWare ou Solaris avant de l'installer réellement sur une machine physique.

1.2.2. CentOS 7 :

CentOS veut dire Community entreprise Operating System, c'est une distribution GNU/Linux principalement destinée aux serveurs. Utilisé par plus de 20% des serveurs web linux, elle est l'une des distributions Linux les plus populaires pour les serveurs web. [9]

1.2.3. HAproxy :

HAProxy est un logiciel libre de répartition de charge écrit par Willy Tarreau. Il est disponible sous Unix (on trouve très facilement des binaires précompilés pour Linux et Solaris).

HAProxy peut aider à mettre en place des solutions de hautes disponibilités, de répartition de charge et de proxy pour tous les types de protocole (TCP ou HTTP).

Il est particulièrement bien adapté pour la gestion des très fortes charges sur un site Web tout en gérant la notion de persistance ou la manipulation des données de la couche 7 du Modèle OSI. [8]

1.2.4. Apache Tomcat :

Apache Tomcat est un conteneur web libre de servlets et JSP. Issu du projet Jakarta, c'est un des nombreux projets de l'Apache Software Foundation.

Il implémente les spécifications des servlets et des JSP du Java Community Process , est paramétrable par des fichiers XML et des propriétés, et inclut des outils pour la configuration et la gestion. [13]

Il comporte également un serveur HTTP.

1.2.5. Percona Server for MySQL :

Est un système de gestion de base de données relationnelle créer par Percona, est un SGBDR open source. Il s'agit d'une solution entièrement compatible pour Oracle MySQL.

Le logiciel inclut un certain nombre de fonctionnalités d'évolutivité, de disponibilité, de sécurité et de sauvegarde uniquement disponibles dans l'édition commerciale de MySQL. [14]

1.2.6. GlusterFS :

GlusterFS est un système de fichiers libre distribué en parallèle, qui permet de stocker jusqu'à plusieurs pétaoctets (10^{15} octets). C'est un système de fichiers de cluster.

RedHat est le mainteneur principal de GlusterFS depuis l'acquisition de la compagnie Gluster en octobre 2011. Il a été d'abord diffusé sous le nom de 'Red Hat Storage Server', puis renommé en 2015 'Red Hat Gluster Storage'. [11]

1.2.7. Varnish :

Varnish est un serveur de cache HTTP apparu en 2006 et distribué sous licence BSD.

Déployé en tant que proxy inverse entre les serveurs d'applications et les clients, il permet de décharger les premiers en mettant en cache leurs données, selon des règles définies par l'administrateur système et les développeurs du site, pour servir plus rapidement les requêtes, tout en allégeant la charge des serveurs. [16]

Avant de commencer la configuration des déférents logicielles, on crée plusieurs machine virtuelles sur le lap top déjà situé dans les 'outils matérielles' avec le logiciel « VMware Workstation Pro », et on installe au sein de chacun machine le système d'exploitation « CentOS 7 », ensuite on installe et configure les logicielles en eux comme suit :

(02) VMs → (02) Serveurs HAproxy + (02) Serveurs Varnish.

- ✓ **@IP Virtuelle : 192.168.112.180**
- ✓ **HAProxy 1: 192.168.112.176**
- ✓ **HAProxy 2 : 192.168.112.178**

(02) VMs → (02) Serveurs Apache Tomcat.

- ✓ **Tomcat 1 : 192.168.112.177**
- ✓ **Tomcat 2 : 192.168.112.174**

(02) VMs → (02) Serveurs Percona Server for MySQL.

- ✓ **Percona 1 : 192.168.112.175**
- ✓ **Percona 2 : 192.168.112.179**

(02) VMs → (02) Serveurs GlusterFS.

- ✓ **GlusterFS1 : 192.168.112.185**
- ✓ **GlusterFS 2 : 192.168.112.186**

2. Processus de configuration:

2.1. Manuel d'installation et configuration de haproxy :

Installation des logiciels nécessaires :

sudo yum update -y
sudo yum install gcc pcre-static pcre-devel openssl-devel.x86_64 –y

Téléchargement du code source de la version 1.7.9 du logiciel haproxy :

cd /opt
wget http://www.haproxy.org/download/1.9/src/haproxy-1.9.7.tar.gz

Décompression du fichier :

tar -xzvf /opt/haproxy-1.9.7.tar.gz && cd /opt/haproxy-1.9.7

Compilation et installation du logiciel :

Compilation :

make USE_OPENSSL=1 USE_ZLIB=1 USE_PCRE=1 TARGET=custom CPU=86x64 USE_LINUX_SPLICE=1 USE_LINUX_TPROXY=1 USE_CPU_AFFINITY=1

USE_OPENSSL=1 # pour activer la gestion du SSL
USE_ZLIB=1 # pour activer la compression
USE_PCRE=1 # utilisation des procédures PERL spécifiques
TARGET=custom # compilation avec un noyau spécifique
CPU=86x64 # Utilisation de l'architecture système 86x64
USE_LIBCRYPT=1 # Utilisation des scripts de cryptage
USE_LINUX_TPROXY=1 # Utilisation de l'option TPROXY pour permettre le transfert de l'adresse IP du client vers les serveurs glassFish
USE_CPU_AFFINITY=1 # pour active la gestion du multi processing. [11]

Installation :

make install

Copie du fichier exécutable vers le dossier bin du système :

cp /usr/local/sbin/haproxy* /usr/sbin/

Création du service haproxy :

```
cp examples/haproxy.init /etc/init.d/haproxy
chmod +x /etc/init.d/haproxy
mkdir -p /etc/haproxy
mkdir -p /var/lib/haproxy
touch /var/lib/haproxy/stats
service haproxy check
service haproxy start
```

Ajout de l'utilisateur haproxy :

```
sudo useradd -r haproxy
```

Paramétrage du pare-feu pour permettre les deux services http et https et ouverture des ports de statistiques :

```
sudo firewall-cmd --permanent --zone=public --add-service=http
sudo firewall-cmd --permanent --zone=public --add-service=https
sudo firewall-cmd --permanent --zone=public --add-port=3001/tcp
sudo firewall-cmd --permanent --zone=public --add-port=3002/tcp
sudo firewall-cmd --permanent --zone=public --add-port=3003/tcp
sudo firewall-cmd --permanent --zone=public --add-port=3004/tcp
sudo firewall-cmd –reload
```

Installation de Keepalived et Corosync Pacemaker sur HAProxy :

Installation et configuration de KeepAlived sur Haproxy :

```
# apt-get install haproxy
# apt-get install keepalived

net.ipv4.ip_nonlocal_bind=1

-vim /etc/sysctl.conf

-net.ipv4.ip_nonlocal_bind=1

-sysctl –p
net.ipv4.ip_nonlocal_bind = 1

-vim /etc/keepalived/keepalived.conf
```

Fichier de configuration maitre :

```
global_defs {
# Keepalived process identifier
lvs_id haproxy_DH
}
# Script used to check if HAProxy is running
vrrp_script check_haproxy {
script "killall -0 haproxy"
interval 2
weight 2
}
# Virtual interface
# The priority specifies the order in which the assigned interface to take over in a failover
vrrp_instance VI_01 {
state MASTER
interface eth0
virtual_router_id 51
priority 101
# The virtual ip address shared between the two loadbalancers
virtual_ipaddress {
192.168.112.180
```

```
 }
 track_script {
check_haproxy
}
}
```
[12]

Fichier de configuration esclave :

```
 global_defs {
# Keepalived process identifier
 lvs_id haproxy_DH_passive
 }
# Script used to check if HAProxy is running
vrrp_script check_haproxy {
script "killall -0 haproxy"
interval 2
weight 2
}
# Virtual interface
# The priority specifies the order in which the assigned interface to take over in a failover
vrrp_instance VI_01 {
state SLAVE
interface eth0
virtual_router_id 51
priority 100
# The virtual ip address shared between the two loadbalancers
virtual_ipaddress {
192.168.112.180
}
track_script {
check_haproxy
}
}
```

-service keepalived start

-vim /etc/default/haproxy

-vim /etc/haproxy/haproxy.cfg

-global

log 127.0.0.1 local0

log 127.0.0.1 local1 notice

#log loghost local0 info

maxconn 4096

#chroot /usr/share/haproxy

user haproxy

group haproxy

daemon

#debug

#quiet

defaults

log global

mode http

option httplog

option dontlognull

retries 3

option redispatch

maxconn 2000

contimeout 5000

clitimeout 50000

srvtimeout 50000

listen stats 192.168.6.ABC:8989

mode http

stats enable

stats uri /stats

stats realm HAProxy\ Statistics

stats auth admin:admin

listen am_cluster 0.0.0.0:80

mode http

balance roundrobin

option httpclose

option forwardfor

cookie SERVERNAME insert indirect nocache

server am-1 192.168.X.ABC:80 cookie s1 check

server am-2 192.168.Y.ABC:80 cookie s2 check

Instalation et configuration de corocync pacemaker :

Définition des addresses IPs et les ajouter à /etc/hosts :

192.169.112.176/24 haproxy01
192.168.112.178/24 haproxy02

Désactiver firewalld :

systemctl stop firewalld.service
systemctl mask firewalld.service

Désactiver SELinux :

setenforce 0

Editer /etc/selinux/config :

SELINUX=permissive

Ajouter à /etc/sysctl.d/haproxy.conf :

net.ipv4.ip_nonlocal_bind = 1

Installer les packages :

yum install pacemaker corosync haproxy pcs fence-agents-all

pcsd est en charge de la synchronisation de la configuration du cluster sur les nœuds :

passwd hacluster

systemctl enable pcsd.service pacemaker.service corosync.service haproxy.service

systemctl start pcsd.service

pcs cluster auth haproxy01 haproxy02

pcs cluster setup --start --name http-cluster haproxy01 haproxy02

pcs cluster enable –all

Vérification :

corosync-cfgtool -s

corosync-cmapctl | grep members

pcs status corosync

pcs status

pcs property set stonith-enabled=false

pcs property set no-quorum-policy=ignore

Pour empêcher la défaillance d'une ressource lors de la récupération d'un nœud :

pcs resource defaults resource-stickiness=100

crm_verify -L –V

J'ajoute une resource:

pcs resource create ClusterIP-01 ocf:heartbeat:IPaddr2 ip=10.0.0.3 cidr_netmask=24 op monitor interval=5s

pcs resource create ClusterIP-02 ocf:heartbeat:IPaddr2 ip=10.0.0.4 cidr_netmask=24 op monitor interval=5s

pcs resource create HAproxy systemd:haproxy op monitor interval=5s

Nous regroupons les addresses IPs ensemble :

pcs resource group add HAproxyIPs Haproxy01 Haproxy02

Ajoutez des contraintes pour déplacer les adresses IP vers l'autre hôte lorsque HAproxy est hors service :

pcs constraint colocation add HAproxy HAproxyIPs INFINITY

pcs constraint order HAproxyIPs then HAproxy

Enfin, j'ai configuré HAproxy avec les deux applications Web et des backends différents, avec http et https. Ce fichier /etc/haproxy/haproxy.cnf doit être identique sur les deux serveurs :

```
global
 log         127.0.0.1 local2
 chroot      /var/lib/haproxy
 pidfile     /var/run/haproxy.pid
 maxconn     100000
 user        haproxy
 group       haproxy
 daemon
 stats socket /var/lib/haproxy/stats
 ssl-server-verify none
 tune.ssl.default-dh-param 2048
defaults
 log                 global
 mode                http
 option              httplog
```

```
    option                  dontlognull
    option                  redispatch
    option forwardfor       except 127.0.0.0/8
    option http-server-close
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           5s
    maxconn                 50000
peers ha-web
    peer haproxy1 192.168.112.176:1024
    peer haproxy2 192.168.112.178:1024

# Frontend servers
listen admin
    bind *:8080
    stats enable
frontend web-http
    bind *:80
    default_backend apache-80
frontend www1-https
    bind 192.168.112.177:443 ssl crt /etc/haproxy/www1.pem
    reqadd X-Forwarded-Proto: https
    default_backend apache-www1-443
frontend laboratorios-https
    bind 192.168.112.174:443 ssl crt /etc/haproxy/www2.pem
    reqadd X-Forwarded-Proto: https
    default_backend apache-www2-443
# Backend servers
```

```
backend apache-80
    stick-table type ip size 20k peers ha-web
    stick on src
    balance roundrobin
    option httpchk GET /server-status
    fullconn    10000
    server tomcat1 192.168.112.177:80 check maxconn 5000
    server tomcat2 192.168.112.174:80 check maxconn 5000
backend apache-www1-443
    stick-table type ip size 20k peers ha-web
    stick on src
    balance roundrobin
    #option ssl-hello-chk
    option httpchk GET /server-status
    fullconn    10000
    server tomcat1 10.0.10.1:443 check port 80 ssl verify none maxconn 5000
    server tomcat2 10.0.10.2:443 check port 80 ssl verify none maxconn 5000
backend apache-www2-443
    stick-table type ip size 20k peers ha-web
    stick on src
    balance roundrobin
    #option ssl-hello-chk
    option httpchk GET /server-status
    fullconn    10000
    server tomcat1 192.168.112.177:443 check port 80 ssl verify none maxconn 5000
    server tomcat2 192.168.112.174:443 check port 80 ssl verify none maxconn 5000
```

2.2. Installation et configuration de Tomcat:

Installer OpenJDK :

Tomcat 8.5 nécessite Java SE 7 ou une version ultérieure. Dans ce didacticiel, nous installerons OpenJDK 8, l'implémentation open source de la plate-forme Java, qui est le développement et l'exécution Java par défaut dans CentOS 7. [13]

L'installation est simple et directe :

- yum install java-1.8.0-openjdk-devel

Créer un utilisateur système Tomcat :

Exécuter Tomcat en tant qu'utilisateur root pose un risque pour la sécurité et n'est pas recommandé. Au lieu de cela, nous allons créer un nouvel utilisateur système et un groupe avec le répertoire personnel / opt / tomcat qui exécutera le service Tomcat:

- sudo useradd -m -U -d /opt/tomcat -s /bin/false tomcat

Télécharger Tomcat :

Nous allons télécharger la dernière version de Tomcat 8.5.x à partir de la page de téléchargement de Tomcat. Au moment de la rédaction de cet article, la dernière version est la 8.5.31. Avant de passer à l'étape suivante, vérifiez la nouvelle version de la page de téléchargement.

Accédez au répertoire / tmp et utilisez wget pour télécharger le fichier zip :

-cd /tmpwget http://www-us.apache.org/dist/tomcat/tomcat-8/v8.5.37/bin/apache-tomcat-8.5.37.zip

Une fois le téléchargement terminé, extrayez le fichier zip et déplacez-le dans le répertoire / opt / tomcat :

- unzip apache-tomcat-*.zip
-sudo mkdir -p /opt/tomcat
-sudo mv apache-tomcat-8.5.37 /opt/tomcat/

L'utilisateur tomcat que nous avons précédemment configuré doit avoir accès au répertoire tomcat. Changez la propriété du répertoire en utilisateur et groupe tomcat :

-sudo chown -R tomcat: /opt/tomcat

Rendez les scripts du répertoire bin exécutables :

-sudo sh -c 'chmod +x /opt/tomcat/latest/bin/*.sh'

Créer un fichier unité system :

Pour exécuter Tomcat en tant que service, créez un fichier d'unité tomcat.service dans le répertoire / etc / systemd / system / avec le contenu suivant :

```
[Unit]
Description=Tomcat 8.5 servlet container
After=network.target
[Service]
Type=forking
User=tomcat
```

```
Group=tomcat
Environment="JAVA_HOME=/usr/lib/jvm/jre"
Environment="JAVA_OPTS=-Djava.security.egd=file:///dev/urandom"
Environment="CATALINA_BASE=/opt/tomcat/latest"
Environment="CATALINA_HOME=/opt/tomcat/latest"
Environment="CATALINA_PID=/opt/tomcat/latest/temp/tomcat.pid"
Environment="CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC"
ExecStart=/opt/tomcat/latest/bin/startup.sh
ExecStop=/opt/tomcat/latest/bin/shutdown.sh
[Install]
WantedBy=multi-user.target
```

Indiquez à systemd que nous avons créé un nouveau fichier unité et démarrez le service Tomcat en exécutant :

```
sudo systemctl daemon-reload
sudo systemctl start tomcat
```

Vérifiez l'état du service avec la commande suivante :

```
sudo systemctl status tomcat
```

S'il n'y a pas d'erreur, vous pouvez activer le démarrage automatique du service Tomcat au démarrage :

```
sudo systemctl enable tomcat
```

Ajuster le firewall :

Si votre serveur est protégé par un pare-feu et que vous souhaitez accéder à l'interface tomcat de l'extérieur du réseau local, ouvrez le port 8080.

Utilisez les commandes suivantes pour ouvrir le port nécessaire :

```
sudo firewall-cmd --zone=public --permanent --add-port=8080/tcp
sudo firewall-cmd –reload
```

Configurer l'interface de gestion Web Tomcat :

À ce stade, Tomcat est installé et nous pouvons y accéder avec un navigateur Web sur le port 8080, mais nous ne pouvons pas accéder à l'interface de gestion Web car nous n'avons pas encore créé d'utilisateur.

Les utilisateurs Tomcat et leurs rôles sont définis dans le fichier tomcat-users.xml.

Si vous ouvrez le fichier, vous remarquerez qu'il est rempli de commentaires et d'exemples décrivant comment le configurer.

sudo nano /opt/tomcat/latest/conf/tomcat-users.xml

Pour ajouter un nouvel utilisateur pouvant accéder à l'interface Web de tomcat (manager-gui et admin-gui), nous devons définir l'utilisateur dans le fichier tomcat-users.xml comme indiqué ci-dessous. Assurez-vous de changer le nom d'utilisateur et le mot de passe pour quelque chose de plus sécurisé: [13]

```
<tomcat-users>
  <role rolename="admin-gui"/>
  <role rolename="manager-gui"/>
  <user username="Tomcat" password="rooting" roles="admin-gui,manager-gui"/>
</tomcat-users>
```

Par défaut, l'interface de gestion Web de Tomcat est configurée pour autoriser l'accès uniquement à partir de l'hôte local. Si vous souhaitez pouvoir accéder à l'interface Web à partir d'une adresse IP distante ou de n'importe quel emplacement non recommandé car dangereux pour la sécurité, vous pouvez ouvrir les fichiers suivants et apporter les modifications suivantes.

Si vous devez accéder à l'interface Web de n'importe où, ouvrez les fichiers suivants et commentez ou supprimez les lignes en commentaires :

/opt/tomcat/latest/webapps/manager/META-INF/context.xml

```
<Context antiResourceLocking="false" privileged="true" >
<!--
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />
-->
</Context>
```

/opt/tomcat/latest/webapps/host-manager/META-INF/context.xml

```
<Context antiResourceLocking="false" privileged="true" >
<!--
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />
-->
```

```
</Context>
```

Si vous devez accéder à l'interface Web uniquement à partir d'une adresse IP spécifique, ajoutez votre adresse IP publique à la liste au lieu de commenter les blocs. Supposons que votre adresse IP publique est 192 .168.112.100 et que vous souhaitez autoriser l'accès uniquement à partir de cette adresse IP :

Vim /opt/tomcat/latest/webapps/manager/META-INF/context.xml

```
<Context antiResourceLocking="false" privileged="true" >
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1|192.168.112.100" />
</Context>
```

Vim /opt/tomcat/latest/webapps/host-manager/META-INF/context.xml

```
<Context antiResourceLocking="false" privileged="true" >
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1|192.168.112.100" />
</Context>
```

La liste des adresses IP autorisées est une liste séparée par une barre verticale |. Vous pouvez ajouter des adresses IP uniques ou utiliser des expressions régulières.

Redémarrez le service Tomcat pour que les modifications prennent effet :

```
sudo systemctl restart tomcat
```

2.3. Installation et configuration Percona for MySQL :

Installation à partir du référentiel Percona :

Installation des packages du cluster Percona XtraDB :

-yum install Percona-XtraDB-Cluster-57

Démarrez le serveur de cluster Percona XtraDB :

Service mysql start

Copiez le mot de passe temporaire généré automatiquement pour le compte superutilisateur

sudo grep 'temporary password' /var/log/mysqld.log

Utilisez ce mot de passe pour vous connecter en tant que root :

mysql -u root –p

Modifiez le mot de passe du compte superuser et déconnectez-vous :

mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'rooting';

Query OK, 0 rows affected (0.00 sec)

mysql> exit

Arrêtez le service mysql :

Service mysql stop

Configuration du premier nœud :

Assurez-vous que le fichier de configuration /etc/my.cnf sur le premier noeud (percona1) contient les éléments suivants :

```
[mysqld]
datadir=/var/lib/mysql
user=mysql
# Path to Galera library
wsrep_provider=/usr/lib64/libgalera_smm.so
# Cluster connection URL contains the IPs of node#1 and node#2
wsrep_cluster_address=gcomm://192.168.112.175,192.168.112.179
# In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB
```

```
# This InnoDB autoincrement locking mode is a requirement for Galera
innodb_autoinc_lock_mode=2
# Node 1 address
wsrep_node_address=192.168.112.175
# SST method
wsrep_sst_method=xtrabackup-v2
# Cluster name
wsrep_cluster_name=perconacluster
# Authentication for SST method
wsrep_sst_auth="sstuser:s3cret"
```

Démarrez le premier noeud avec la commande suivante :

```
systemctl start mysql@bootstrap.service
```

Il n'est pas recommandé de laisser un mot de passe vide pour le compte root. Le mot de passe peut être changé comme suit :

```
mysql@percona1> UPDATE mysql.user SET password=rooting where user='root';
mysql@percona1> FLUSH PRIVILEGES;
```

Pour effectuer un transfert d'instantané d'état à l'aide de XtraBackup, configurez un nouvel utilisateur avec les privilèges appropriés :

```
mysql@percona1> CREATE USER 'sstuser'@'localhost' IDENTIFIED BY 's3cret';
mysql@percona1> GRANT PROCESS, RELOAD, LOCK TABLES, REPLICATION CLIENT ON *.* TO 'sstuser'@'localhost';
mysql@percona1> FLUSH PRIVILEGES;
```

Configuration du deuxième nœud :

Assurez-vous que le fichier de configuration /etc/my.cnf sur le deuxième noeud (percona 2) contient les éléments suivants :

```
[mysqld]
datadir=/var/lib/mysql
user=mysql
# Path to Galera library
wsrep_provider=/usr/lib64/libgalera_smm.so

# Cluster connection URL contains IPs of node#1 and node#2
wsrep_cluster_address=gcomm://192.168.112.175,192.168.172.179
# In order for Galera to work correctly binlog format should be ROW
```

```
binlog_format=ROW
# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB
# This InnoDB autoincrement locking mode is a requirement for Galera
innodb_autoinc_lock_mode=2
# Node 2 address
wsrep_node_address=192.168.172.179
# Cluster name
wsrep_cluster_name=perconacluster
# SST method
wsrep_sst_method=xtrabackup-v2
#Authentication for SST method
wsrep_sst_auth="sstuser:s3cret"
```

Démarrez le deuxième noeud avec la commande suivante :

```
/etc/init.d/mysql start
```

[14]

2.4. Installation et configuration de GlusterFS :

Utilisation des packages Gluster.org :

yum update –y

Téléchargez le dernier référentiel glusterfs-epel à partir de gluster.org :

yum install wget -y

wget -P /etc/yum.repos.d/ http://download.gluster.org/pub/gluster/glusterfs/LATEST/CentOS/glusterfs-epel.repo [15]

Installez également le dernier référentiel EPEL de fedoraproject.org pour résoudre toutes les dépendances :

yum install http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

Les deux référentiels sont activés par défaut :

yum repolist

Installez les packages GlusterFS Server et Samba sur les deux nœuds de cluster de stockage :

yum install glusterfs-server samba –y

Créez un nouveau volume physique à l'aide du disque / dev / vdb :

pvcreate /dev/vdb

Créez un groupe de volumes sur / dev / vdb :

vgcreate vg_gluster /dev/vdb

Créez des volumes logiques brick1 et brick2 pour les briques XFS sur les deux nœuds de cluster :

lvcreate -L 5G -n brick1 vg_gluster

lvcreate -L 5G -n brick2 vg_gluster

Configurez les systèmes de fichiers XFS :

mkfs.xfs /dev/vg_gluster/brick1

mkfs.xfs /dev/vg_gluster/brick2

Créez des points de montage et des briques XFS :

mkdir -p /bricks/brick{1,2}

mount /dev/vg_gluster/brick1 /bricks/brick1

mount /dev/vg_gluster/brick2 /bricks/brick2

Activez et démarrez glusterfsd.service sur les deux nœuds :

```
# systemctl enable glusterd.service

# systemctl start glusterd.service
```

Activer les ports requis sur le pare-feu :

```
# firewall-cmd --zone=public --add-port=24007-24008/tcp --permanent

# firewall-cmd –reload
```

Utilisez la commande gluster pour connecter le deuxième nœud GlusterFS et créer un pool approuvé (cluster de stockage) :

```
# gluster peer probe gluster2
```

Vérifier l'homologue du cluster :

```
# gluster peer status
```

Ouvrez le port requis sur le pare-feu :

```
# firewall-cmd --zone=public --add-port=24009/tcp --permanent

# firewall-cmd –reload
```

Utilisez la partition XFS / bricks / brick1 sur les deux nœuds pour créer un volume répliqué hautement disponible. Commencez par créer un sous-répertoire dans / bricks / brick1 point de montage. Ce sera nécessaire pour GlusterFS :

```
# mkdir /bricks/brick1/brick
```

Créez un volume GlusterFS répliqué :

```
# gluster volume create glustervol1 replica 2 transport tcp gluster1:/bricks/brick1/brick \

gluster2:/bricks/brick1/brick

# gluster volume start glustervol1
```

Vérifiez les volumes GlusterFS :

```
# gluster volume info all
```

Ouvrez le pare-feu pour les clients Glusterfs / NFS / CIFS :

firewall-cmd --zone=public --add-service=nfs --add-service=samba --add-service=samba-client --permanent

firewall-cmd --zone=public --add-port=111/tcp --add-port=139/tcp --add-port=445/tcp --add-port=965/tcp --add-port=2049/tcp \

--add-port=38465-38469/tcp --add-port=631/tcp --add-port=111/udp --add-port=963/udp --add-port=49152-49251/tcp --permanent

firewall-cmd –reload

Installez les packages du client GlusterFS :

yum install glusterfs glusterfs-fuse attr –y

Monter les volumes GlusterFS sur le client :

mount -t glusterfs gluster1:/glustervol1 /mnt/

Ajoutez une nouvelle ligne au fichier / etc / fstab (facultatif) :

gluster1:/glustervol1 /mnt glusterfs defaults,_netdev 0 0

Sur les deux nœuds, ajoutez la ligne suivante au fichier /etc/nfsmount.conf :

Defaultvers=3

Mount GlusterFS Volumes via NFS :

mount -t nfs gluster1:/glustervol1 /mnt/

Ajoutez la ligne suivante à / etc / fstab (facultatif) :

gluster1 :/glustervol1 /mnt nfs defaults,_netdev 0 0 [15]

2.5. Installation du serveur de cache Varnish :

Il existe maintenant des packages RPM pré-compilés pour la dernière version de Varnish Cache 5 (c'est-à-dire la version 5.2 au moment de la rédaction), vous devez donc ajouter un référentiel officiel de Varnish Cache.

Avant cela, vous devez activer le référentiel EPEL pour installer plusieurs packages de dépendance, comme indiqué :

```
# yum install -y epel-release
```

Ensuite, installez pygpgme, un package de traitement des signatures GPG et yum-utils, un ensemble d'utilitaires utiles qui étendent les fonctionnalités natives de yum de différentes façons :

```
# yum install pygpgme yum-utils
```

Créez maintenant un fichier nommé /etc/yum.repos.d/varnishcache_varnish5.repo qui contient la configuration du référentiel ci-dessous :

```
# vim /etc/yum.repos.d/varnishcache_varnish5.repo
[varnishcache_varnish5]
name=varnishcache_varnish5
baseurl=https://packagecloud.io/varnishcache/varnish5/centos/7/$basearch
repo_gpgcheck=1
gpgcheck=0
enabled=1
gpgkey=https://packagecloud.io/varnishcache/varnish5/gpgkey
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300
[varnishcache_varnish5-source]
name=varnishcache_varnish5-source
baseurl=https://packagecloud.io/varnishcache/varnish5/centos/7/SRPMS
repo_gpgcheck=1
gpgcheck=0
enabled=1
gpgkey=https://packagecloud.io/varnishcache/varnish5/gpgkey
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300
```

[16]

Exécutez maintenant la commande ci-dessous pour mettre à jour votre cache yum local et installer le package varnish cache 5 (n'oubliez pas d'accepter la clé GPG en tapant y ou yes lors de l'installation du package) :

```
# yum -q makecache -y --disablerepo='*' --enablerepo='varnishcache_varnish5'
```

```
# yum install varnish
```

Après avoir installé Varnish Cache, l'exécutable principal sera installé en tant que / usr / sbin / varnishd et les fichiers de configuration de varnish se trouvent dans / etc / varnish / :

/etc/varnish/varnish.params - il s'agit du fichier de configuration de l'environnement varnish.

/etc/varnish/default.vcl - il s'agit du fichier de configuration principal du varnish. Il est écrit à l'aide du langage de configuration VCL (varnish configuration language).

/ etc / varnish / secret - fichier secret de varnish.

Vous pouvez vérifier que l'installation de Varnish a réussi en consultant l'emplacement de l'exécutable et la version de Varnish installée sur votre système.

```
$ which varnishd
```

```
$ varnishd –V
```

Configurez maintenant Apache pour fonctionner avec Varnish Cache. Par défaut, Apache écoute sur le port 80, vous devez modifier le port HTTPD par défaut en 8080 - cela garantira que HTTPD s'exécutera derrière la mise en cache Varnish.

Vous pouvez utiliser la commande sed pour modifier le port 80 en 8080, comme indiqué.

```
# sed -i "s/Listen 80/Listen 8080/" /etc/httpd/conf/httpd.conf
```

Ensuite, ouvrez le fichier de configuration de l'environnement varnish et recherchez le paramètre VARNISH_LISTEN_PORT qui spécifie le port sur lequel Varnish est à l'écoute et modifiez sa valeur de 6081 à 80.

```
# vi /etc/varnish/varnish.params
```

Ensuite, configurez Apache en tant que serveur principal pour le proxy Varnish, dans le fichier de configuration /etc/varnish/default.vcl :

```
# vi /etc/varnish/default.vcl
```

Recherchez la section dorsale et définissez l'adresse IP et le port de l'hôte. La configuration d'arrière-plan par défaut ci-dessous, configurez-la de manière à pointer vers votre serveur de contenu actuel :

```
backend default {
  .host = "127.0.0.1";
  .port = "8080";
}
```

Après avoir effectué toutes les configurations nécessaires, redémarrez les caches HTTPD et Varnish pour appliquer les modifications ci-dessus :

```
# systemctl restart httpd

# systemctl start varnish

# systemctl enable varnish

 # systemctl status varnish
```

Voici quelques captures d'écran sur la solution :

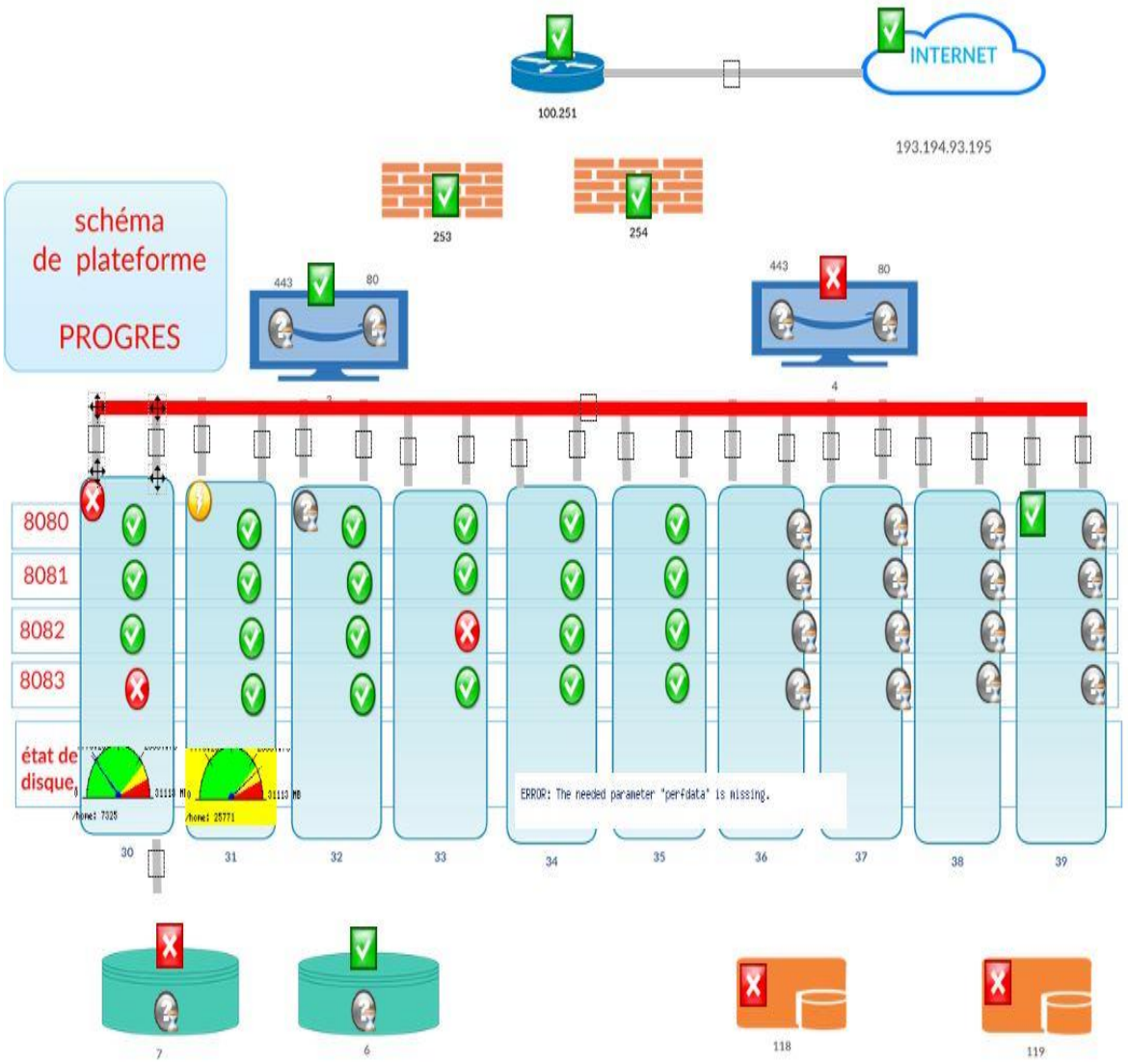


Figure 8 : Schéma de la plateforme PROGRES SUR MESRS. [17]

HAProxy version 1.5.18, released 2016/05/10

Statistics Report for pid 13705

> General process information

pid = 13705 (process #1, nbproc = 16)
uptime = 0d 0h52m56s
system limits: memmax = unlimited; ulimit-n = 800210
maxsock = 800210; **maxconn** = 400000; **maxpipes** = 0
current conns = 174; current pipes = 0/0; conn rate = 2/sec
Running tasks: 1/396; idle = 99 %

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
backup UP
backup UP, going down
backup DOWN, going up
not checked
Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
- Scope :
- Hide 'DOWN' servers
- Disable refresh
- Refresh now
- CSV export

External resources:
- Primary site
- Updates (v1.5)
- Online manual

maintenance

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	400 000	0			0	0	0	0	0					OPEN								

http_front

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	1	-	0	3	400 000	29			9 419	4 652	0	0	12					OPEN								

localhost

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				1	6	-	172	176	400 000	2 206			47 331 548	226 597 933	0	0	737					OPEN								

webreferentiel-backend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Cluster_ref_20_tomcat1	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	52m55s UP	L7OK/200 in 0ms	1	Y	-	0	0	0s	-
Cluster_ref_20_tomcat2	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	52m55s UP	L7OK/200 in 0ms	1	Y	-	0	0	0s	-
Cluster_ref_20_tomcat3	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	52m55s UP	L7OK/200 in 0ms	1	Y	-	0	0	0s	-
Cluster_ref_20_tomcat4	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	52m55s UP	L7OK/200 in 0ms	1	Y	-	0	0	0s	-
Backend	0	0		0	0		0	0	40 000	0	0	?	0	0	0	0		0	0	0	0	52m55s UP		4	4	0		0	0s	

webfve-backend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Cluster_prod_31_tomcat1	0	0	-	0	7		0	4	100	231	6	4m35s	225 928	4 890 418		0		0	0	0	0	3m58s DOWN	* L7TOUT in 3001ms	1	Y	-	13/0	2	12m30s	-
Cluster_prod_31_tomcat2	0	0	-	0	6		0	3	100	181	5	1m20s	284 029	2 928 859		0		0	1	0	0	28s DOWN	L7TOUT in 3000ms	1	Y	-	22/0	6	10m53s	-
Cluster_prod_31_tomcat3	0	0	-	0	5		2	4	100	22	1	45m21s	17 036	187 344		0		0	0	0	0	44m39s DOWN	* L7TOUT in 3000ms	1	Y	-	6/0	2	51m13s	-
Cluster_prod_31_tomcat4	0	0	-	0	2		2	2	100	9	1	44m44s	4 306	124 671		0		0	0	0	0	44m44s DOWN	L7TOUT in 3001ms	1	Y	-	4/0	2	51	

10.10.10.4

Figure 9 : Schéma de statistique de HAproxy

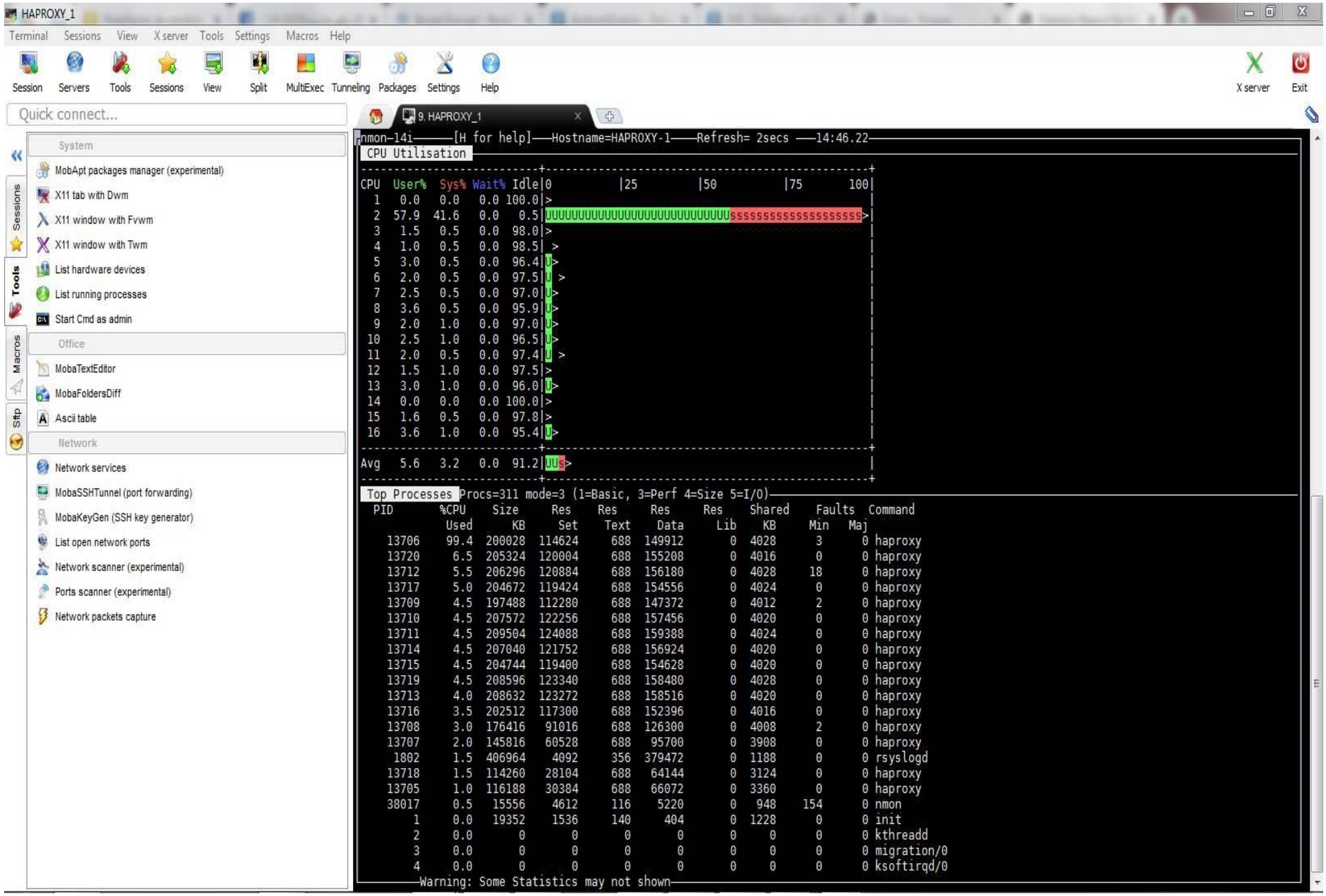


Figure 10 : Schéma de détaille HAproxy _Server

webmaster-backend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server							
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Cluster_prod_34_tomcat1	0	0	-	0	7		0	3	30	242	2	4s	4 215 631	9 675 559		0		0	0	0	0	53m42s UP	L4OK in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_34_tomcat2	0	0	-	0	7		0	3	30	218	2	3m23s	3 947 334	7 933 135		0		0	0	0	0	53m42s UP	L4OK in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_34_tomcat3	0	0	-	0	6		0	5	30	175	2	1m26s	1 749 802	6 938 705		0		0	0	0	0	53m42s UP	L4OK in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_34_tomcat4	0	0	-	0	5		0	4	30	276	2	4m7s	471 259	17 189 264		0		0	0	0	0	53m42s UP	L4OK in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_35_tomcat1	0	0	-	0	1		0	1	30	2	2	12m18s	915	0		0		0	2	0	0	53m42s UP	L4OK in 0ms	1	Y	-	1/0	0	0s	-
Cluster_prod_35_tomcat2	0	0	-	0	1		1	2	30	4	2	1m17s	1 768	0		0		0	3	0	0	53m42s UP	L4OK in 0ms	1	Y	-	10/0	0	0s	-
Cluster_prod_35_tomcat3	0	0	-	0	1		1	1	30	4	2	7s	1 881	0		0		0	3	0	0	53m42s UP	L4OK in 0ms	1	Y	-	7/0	0	0s	-
Cluster_prod_35_tomcat4	0	0	-	0	1		0	1	30	2	1	41m31s	1 284	0		0		0	2	0	0	53m42s UP	L4OK in 0ms	1	Y	-	10/0	0	0s	-
Cluster_prod_36_tomcat1	0	0	-	0	1		0	2	30	3	1	38m40s	2 149	0		0		0	3	0	0	53m28s UP	L4OK in 0ms	1	Y	-	17/0	1	12s	-
Cluster_prod_36_tomcat2	0	0	-	0	6		2	2	30	163	1	4s	2 014 021	8 174 201		0		0	0	0	0	53m42s UP	L4OK in 1ms	1	Y	-	0/0	0	0s	-
Cluster_prod_36_tomcat3	0	0	-	0	6		2	4	30	193	1	28s	3 308 819	8 579 243		0		0	0	0	0	53m42s UP	L4OK in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_36_tomcat4	0	0	-	0	1		0	2	30	3	1	35m45s	1 363	0		0		0	3	0	0	53m26s UP	L4OK in 0ms	1	Y	-	13/0	1	13s	-
Cluster_prod_37_tomcat1	0	0	-	0	6		0	3	30	204	1	4s	635 210	6 869 599		0		0	0	0	0	53m42s UP	L4OK in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_37_tomcat2	0	0	-	0	4		0	4	30	217	1	23s	6 014 544	8 340 871		0		0	0	0	0	53m42s UP	L4OK in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_37_tomcat3	0	0	-	0	5		3	4	30	310	1	11s	9 751 180	12 559 920		0		0	1	0	0	53m42s UP	L4OK in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_37_tomcat4	0	0	-	1	5		0	2	30	141	1	1s	2 769 452	6 708 313		0		0	0	0	0	53m42s UP	L4OK in 2ms	1	Y	-	0/0	0	0s	-
Backend	0	0		1	8		10	14	40 000	2 157	23	1s	34 886 512	92 968 810	0	0		0	17	0	0	53m42s UP		16	16	0		0	0s	

webgrh-backend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server							
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Cluster_prod_31_tomcat1	0	0	-	0	2		0	1	-	16	1	32m35s	10 079	356 466		0		0	1	0	0	48m23s UP	L4OK in 1000ms	1	Y	-	64/0	3	31s	-
Cluster_prod_31_tomcat2	0	0	-	0	2		0	2	-	20	0	3m19s	25 582	122 223		0		0	0	0	0	50m2s UP	L4OK in 0ms	1	Y	-	34/0	1	7s	-
Cluster_prod_31_tomcat3	0	0	-	0	1		0	1	-	4	1	45m50s	3 261	12 212		0		0	0	0	0	50m3s UP	L4OK in 0ms	1	Y	-	77/0	2	14s	-
Cluster_prod_31_tomcat4	0	0	-	0	3		0	1	-	9	1	30m35s	6 063	32 494		0		0	1	0	0	50m3s UP	L4OK in 0ms	1	Y	-	157/0	1	7s	-
Cluster_prod_32_tomcat1	0	0	-	0	1		0	1	-	4	0	35m17s	2 886	12 904		0		0	0	0	0	49m56s UP 2/3 ↓	L4TOUT in 2000ms	1	Y	-	27/0	1	12s	-
Cluster_prod_32_tomcat2	0	0	-	0	1		0	2	-	4	0	21m35s	2 681	81 983		0		0	1	0	0	49m56s UP 2/3 ↓	L4TOUT in 2001ms	1	Y	-	46/0	1	12s	-
Cluster_prod_32_tomcat3	0	0	-	0	2		0	1	-	3	0	48m35s	1 848	110 092		0		0	1	0	0	40m31s UP	L4OK in 0ms	1	Y	-	82/0	2	24s	-
Cluster_prod_32_tomcat4	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	49m56s UP	L4OK in 0ms	1	Y	-	63/0	1	12s	-
Cluster_prod_33_tomcat1	0	0	-	0	3		1	1	-	18	0	14m49s	16 843	157 998		0		0	0	0	0	49m54s UP	*L4OK in 0ms	1	Y	-	30/0	2	19s	-
Cluster_prod_33_tomcat2	0	0	-	0	2		0	1	-	4	0	14m29s	2 414	246 682		0		0	0	0	0	49m54s UP	L4OK in 0ms	1	Y	-	5/0	2	23s	-
Cluster_prod_33_tomcat3	0	0	-	0	1		0	1	-	2	0	46m25s	776	1 381		0		0	1	0	0	49m54s UP	L4OK in 0ms	1	Y	-	27/0	1	12s	-
Cluster_prod_33_tomcat4	0	0	-	0	1		0	1	-	2	0	35m33s	1 117	33 939		0		0	0	0	0	18m31s UP	L4OK in 0ms	1	Y	-	93/0	2	27s	-
Backend	0	0		0	3		1	5	40 000	86	3	3m19s	73 450	1 168 374	0	0		0	5	0	0	50m3s UP		12	12	0		1	3s	

webetrangers-backend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server							
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Cluster_8_tomcat_1	0	0	-	0	1		0	1	-	8	0	12m42s	4 794	321 028		0		0	0	0	0	53m42s UP	L4OK in 0ms	1	Y	-	0	0	0s	-
Cluster_8_tomcat_2	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	53m42s UP	L4OK in 0ms	1	Y	-	0	0	0s	-
Cluster_8_tomcat_3	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	53m42s UP	L4OK in 0ms	1	Y	-	0	0		

10.10.10.4

Figure 11 : Schéma de détaille Web_Master

DOCTORAT-backend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server							
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Cluster_prod_38_tomcat1	0	0	-	0	3		0	3	300	84	4	7m55s	57 744	5 057 033		0		0	0	0	0	54m UP	L7OK/200 in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_38_tomcat2	0	0	-	0	4		0	3	300	140	4	39s	95 865	6 936 731		0		0	0	0	0	54m UP	L7OK/200 in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_38_tomcat3	0	0	-	0	6		0	3	300	170	4	2m22s	122 022	8 702 249		0		0	0	0	0	54m UP	L7OK/200 in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_38_tomcat4	0	0	-	0	4		0	3	300	135	4	12s	92 946	7 181 488		0		0	0	0	0	54m UP	L7OK/200 in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_39_tomcat1	0	0	-	0	3		1	3	300	145	4	1m10s	101 389	7 237 054		0		0	0	0	0	54m UP	L7OK/200 in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_39_tomcat2	0	0	-	0	5		1	2	300	172	4	26s	120 707	6 475 739		0		0	0	0	0	54m UP	L7OK/200 in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_39_tomcat3	0	0	-	0	5		1	3	300	122	4	2m29s	82 744	6 823 623		0		0	0	0	0	54m UP	L7OK/200 in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_39_tomcat4	0	0	-	1	7		1	2	300	154	4	1s	114 985	7 889 917		0		0	0	0	0	54m UP	L7OK/200 in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_50_tomcat1	0	0	-	0	4		1	4	300	171	4	1m21s	126 143	7 956 681		0		0	0	0	0	54m UP	L7OK/200 in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_50_tomcat2	0	0	-	0	4		1	3	300	160	4	32s	108 404	7 439 043		0		0	0	0	0	54m UP	L7OK/200 in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_50_tomcat3	0	0	-	0	5		1	4	300	153	4	2m50s	105 766	7 193 531		0		0	0	0	0	54m UP	L7OK/200 in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_50_tomcat4	0	0	-	0	5		1	3	300	139	4	24s	93 691	8 806 280		0		0	0	0	0	54m UP	L7OK/200 in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_51_tomcat1	0	0	-	0	3		1	3	300	162	4	12s	112 647	8 304 903		0		0	0	0	0	54m UP	L7OK/200 in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_51_tomcat2	0	0	-	0	4		0	2	300	132	4	4s	94 252	6 343 377		0		0	0	0	0	54m UP	L7OK/200 in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_51_tomcat3	0	0	-	0	4		1	2	300	124	4	3s	85 879	6 110 953		0		0	0	0	0	54m UP	L7OK/200 in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_51_tomcat4	0	0	-	0	5		0	6	300	172	4	1m15s	118 255	9 675 622		0		0	0	0	0	54m UP	L7OK/200 in 0ms	1	Y	-	0/0	0	0s	-
Backend	0	0		1	8		10	17	40 000	2 335	64	1s	1 633 439	118 134 224	0	0		0	0	0	0	54m UP		16	16	0		0	0s	

PROD-backend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server							
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Cluster_prod_31_tomcat1	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	37s UP	L4OK in 0ms	1	Y	-	33/0	2	15s	-
Cluster_prod_31_tomcat2	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	50m20s UP	L4OK in 0ms	1	Y	-	8/0	1	7s	-
Cluster_prod_31_tomcat3	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	50m20s UP	L4OK in 0ms	1	Y	-	105/0	2	14s	-
Cluster_prod_31_tomcat4	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	9m9s UP	L4OK in 0ms	1	Y	-	170/0	3	26s	-
Cluster_prod_32_tomcat1	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	7m13s UP	L4OK in 0ms	1	Y	-	114/0	4	50s	-
Cluster_prod_32_tomcat2	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	50m15s UP	L4OK in 0ms	1	Y	-	10/0	1	12s	-
Cluster_prod_32_tomcat3	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	50m15s UP	L4OK in 0ms	1	Y	-	92/0	1	12s	-
Cluster_prod_32_tomcat4	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	50m16s UP	L4OK in 1000ms	1	Y	-	63/0	1	12s	-
Cluster_prod_33_tomcat1	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	50m14s UP	L4OK in 0ms	1	Y	-	52/0	2	19s	-
Cluster_prod_33_tomcat2	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	50m14s UP	L4OK in 0ms	1	Y	-	25/0	2	30s	-
Cluster_prod_33_tomcat3	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	36s UP 2/3 ↓	*L4TOUT in 2001ms	1	Y	-	89/0	8	1m22s	-
Cluster_prod_33_tomcat4	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	50m14s UP	L4OK in 0ms	1	Y	-	80/0	1	12s	-
Cluster_prod_34_tomcat1	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	54m UP	L4OK in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_34_tomcat2	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	54m UP	L4OK in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_34_tomcat3	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	54m UP	L4OK in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_34_tomcat4	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	54m UP	L4OK in 0ms	1	Y	-	0/0	0	0s	-
Cluster_prod_35_tomcat1	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	54m UP	L4OK in 0ms	1	Y	-	2/0	0	0s	-
Cluster_prod_35_tomcat2	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	54m UP	L4OK in 0ms	1	Y	-	18/0	0	0s	-
Cluster_prod_35_tomcat3	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	54m UP	L4OK in 0ms	1	Y	-	5/0	0	0s	-
Cluster_prod_35_tomcat4	0	0	-	0	0		0	0	10	0	0	?	0	0		0		0	0	0	0	54m UP	L4OK in 0ms	1	Y	-	5/0	0	0s	-

10.10.10.4

Figure 12 : Schéma de détaille Web_Doctorat.

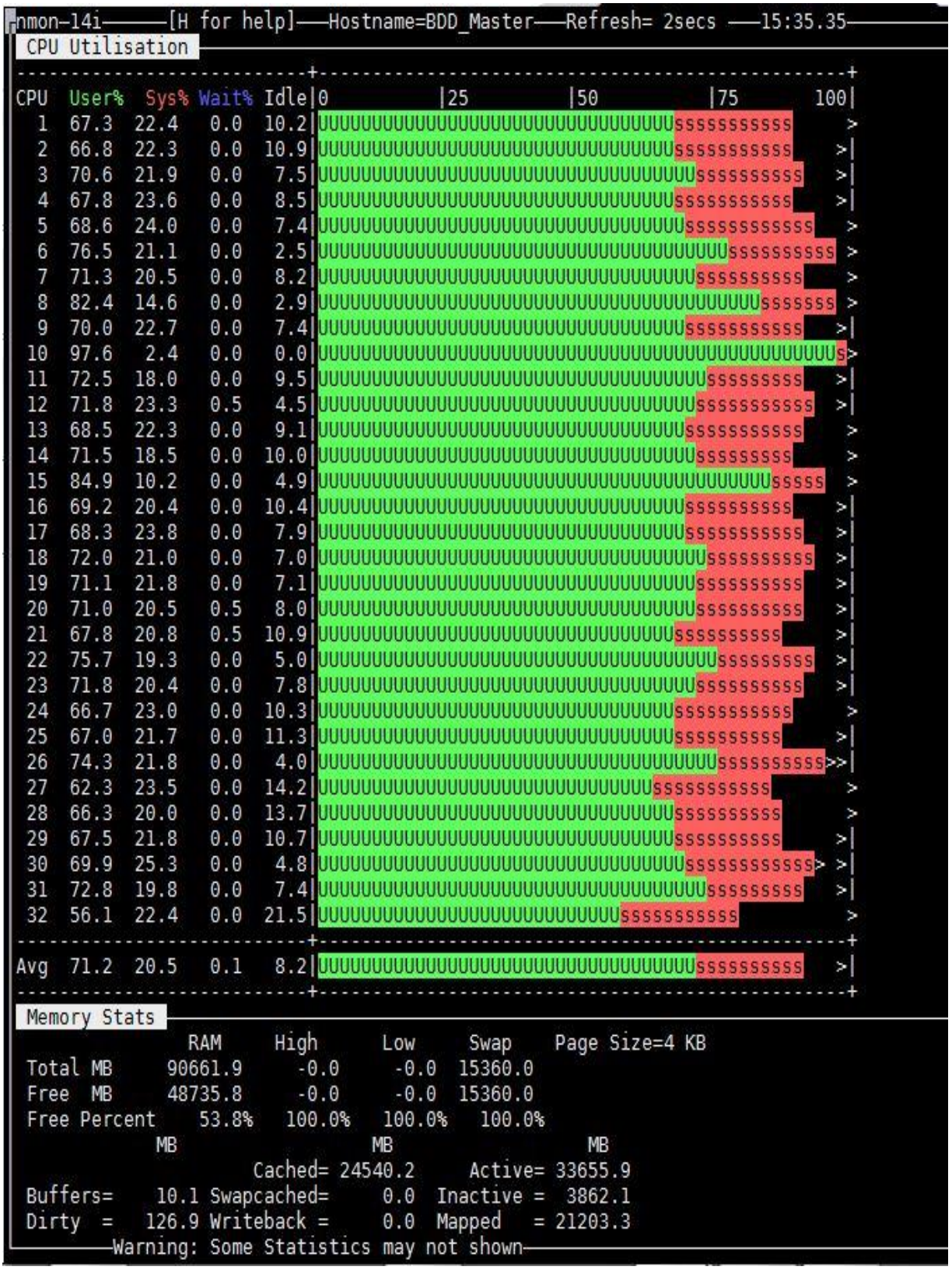


Figure 13 : Schéma de détaille la charge sur la BDD.

Figure 14 : Schéma de détaille google_analytics_des Web_Apps.

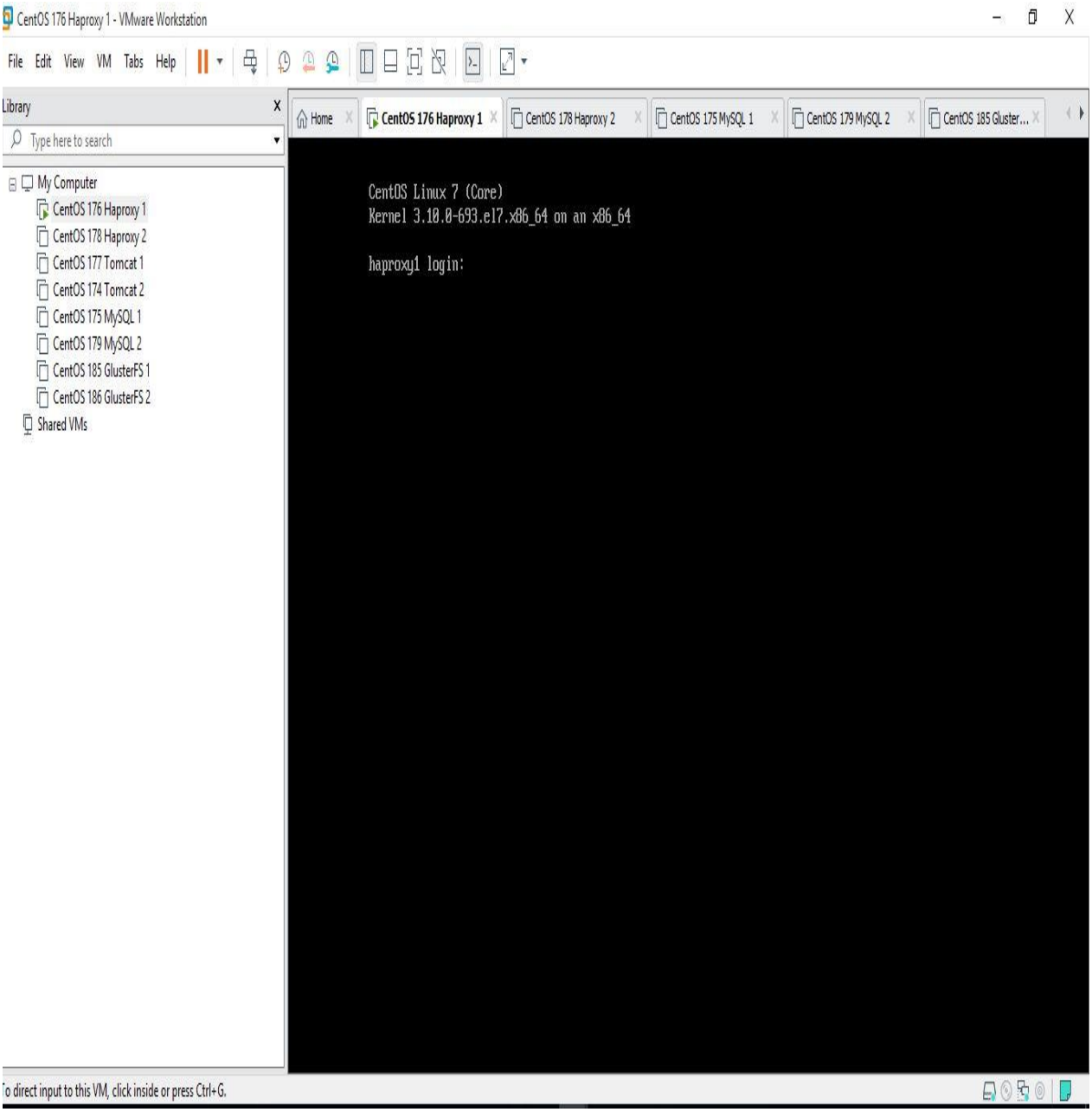


Figure 15 : Schéma de VMs crées sur VMware Workstation Pro .

Conclusion:

Dans ce chapitre nous parlons sur les outils utilisés et les processus de configuration, la solution que j'ai proposée a été testée et validée et mise en production par la Direction des Réseaux et Systèmes d'Information(DRSI) du MESRS.

Conclusion générale et Perspectives

L'objectif de mon projet était de mettre en place une infrastructure de services hautement disponible et répondra nécessairement à deux exigences importantes (capacité et disponibilité des services) et impactera positivement les opérations et le support (moins d'incidents relatifs à la disponibilité et à la capacité).

La solution que j'ai proposée a été validée et mise en production par la Direction des Réseaux et Systèmes d'Information(DRSI) du MESRS.

Enfin, voici quelques perspectives pour améliorer plus notre plateforme en avenir :

- HAProxy a fait ses preuves depuis de nombreuses années, Sa fiabilité et ses performances font de lui un atout majeur dans un cluster de serveurs.

Cette solution présente uniquement une infime partie des possibilités de haute disponibilité.

Il serait intéressant de coupler HAProxy avec Heartbeat qui permet de surveiller la disponibilité des programmes.

Avec ce type de systèmes, nous pourrions combiner de la haute disponibilité matériel et logiciel et ainsi accroître les performances et la stabilité d'un site web.

- Couple HaProxy + Heartbeat = Solution fiable et performante et Améliore la disponibilité des services à moindre coût.

- Extension de l'utilisation d'HaProxy pour d'autres services :
  - Intérêt particulier pour les services d'annuaire (OpenLdap).

- La haute disponibilité est devenue indispensable pour bon nombre de service.

Le matériel devient de plus en plus performant, les capacités et fonctionnalités des composants sont de plus en plus importantes, Les techniques sont de mieux en mieux maitrisées. Alors pourquoi n'atteint on pas la disponibilité de 100%, La technique est une composante mais l'interaction humaine est aussi présente.

Conclusion générale et Perspectives

Les nouvelles technologies et les améliorations incessantes permettent de plus en plus de monter en puissance sans arrêter le service mais il existe toujours des causes non prévisibles.

L'objectif est justement de prévoir ces imprévus et surtout de minimiser leurs causes.

Sur un plan des perspectives, l'augmentation des débits et la disparition des frontières numériques permettra de généraliser cette haute disponibilité. L'objectif serait que ces serveurs applicatifs et de données consomment de moins en moins pour respecter l'environnement.

- Nous envisagerons dans le futur faire des réplications sur des Cloud privé (Cloud computing) pour la dématérialisation des services ou données de l'entreprise.

- L'agitation des développeurs et des sociétés commerciales autour de la haute disponibilité sur Linux et les logiciels libres montre qu'il existe une forte attente dans ce domaine et nous assure d'avoir rapidement des produits adéquats pour apporter à cet environnement des solutions plus performantes que les meilleures offres commerciales disponibles aujourd'hui sur le marché.

Bibliographie :

[1] : ARMEL Francklin et SIMO Tegueu. Plate- forme d'entreprise sécurisée et de haute disponibilité. Licence en ingénierie des réseaux et télécoms, Institut universitaire de technologie FOTSO Victor de Bandjoun, pages 1-27, 2009.

[3] : BOUDAA Boudjemaa. Les communautés pour une haute disponibilité des services Web maintenant la médiation sémantique dans les compositions. PhD thesis, Université Ibn Khaldoun, Tiaret, Algérie, pages 1-84, 2009.

[7] : HADDAD Jugourta et TOUATI Badr Eddine. Tolérance aux pannes des serveurs cas THCIN-LAIT. Master en Informatique, Abderrahmane Mira de Bejaia, Algérie, pages 1-91, 2014.

[8] : KPEGOUNI Abasse. Clustering et haute disponibilité sous Linux. Master professionnelle international en informatique, Université de Lomé Centre Informatique et de Calcul (CIC) et Université de Technologie Belfort , Montbéliard, pages 1-23, 2017.

[10]: Bérenguier, I. (2000). Sécurité des logiciels libres Haute disponibilité. EADS SYCOMBRE, 10(2). 1-34.

[17] : DOCs, Programme d'Appui à la Politique Sectorielle de l'Enseignement Supérieur et de la Recherche Scientifique en Algérie (PAPS ESRS).
Référence : Europeaid/133565/D/SER/DZ - Contrat 319889/2013.

Références bibliographiques

Webographie :

[2] : https://www.celeste.fr/fondamentaux-haute-disponibilite-entreprise

Les fondamentaux de la Haute Disponibilité.

[4] : http://denisrosenkranz.com/tuto-ha-un-cluster-drbdapache-avec-heartbeat-sur-debian-7/

[5] : http://lea-linux.org/documentations/La_haute_disponibilit%C3%A9
Consulté le 29 mai 2016, La Haute Disponibilité.

[6] : http://amineremache.developpez.com/evaluation-cluster/
Consulté le 20 août 2018, Mise en place et évaluation d'un cluster pour l'équilibrage de charge des serveurs Web Par amine4023.

[9] : www.linux-france.org/article/cel/SICOMOR/Haute_disponibilite/html/Rapport_7-2V10.html ,

Consulté le 30 juin 2000, Sécurité des logiciels libres 'Haute disponibilité'.

[11] : https://apuntesderootblog.wordpress.com/2014/08/26/clustered-haproxy-for-load-balancing-web-sites/comment-page-1/

Consulté le 26 août 2014, Clustered HAproxy for load balancing web sites.

[12] : https://dasunhegoda.com/how-to-setup-haproxy-with-keepalived/833/
Consulté le 18 mars 2015, How to setup HAProxy with Keepalived.

[13] : https://www.howtoforge.com/tutorial/how-to-install-tomcat-on-centos/
How to Install Apache Tomcat 8.5 on CentOS 7.3

[14] : https://www.percona.com/doc/perconaxtradcluster/LATEST/howtos/centos_howto.html
How to Install Percona on CentOS 7.

[15] : https://centos.org/HowTos/GlusterFSonCentOS
Consulté le 02 janvier 2017, How to Install GlusterFS on CentOS 7.

[16] : https://www.tecmint.com/install-varnish-cache-on-centos-7-for-apache/
Consulté le 16 novembre 2017, Install Varnish Cache 5.2 to Boost Apache Performance on CentOS 7.

[18] : www.mesrs.dz/

Annexe : Présentation de l'organisme d'accueil

Dans cette annexe nous avons pu situer le projet dans son cadre général en présentant **l'organisme d'accueil** et les butes de la plateforme.

1.1 Présentation du Ministère de l'Enseignement Supérieur et de la Recherche Scientifique (MESRS) :

Le Ministère de l'Enseignement Supérieur et de la recherche scientifique est un établissement gouvernemental fondé pour veiller au déploiement du réseau des établissements publics d'enseignement supérieur à travers le territoire conformément aux objectifs poursuivis par le Gouvernement en matière d'aménagement du territoire et d'égalité d'accès aux cycles de l'enseignement supérieur.

1.2 Les missions et les attributions du MESRS :

Le MESRS a pour mission la formation des cadres nécessaires au développement économique, social et culturel afin de répondre aux besoins en matière de recherche scientifique et technologique.

Le ministre de l'enseignement supérieur et de la recherche scientifique propose les éléments de politique nationale dans le domaine de l'enseignement supérieur et de la recherche scientifique et du **développement technologique, et en assure la mise en œuvre, conformément aux lois et règlements en vigueur**.

Le ministre de l'enseignement supérieur et de la recherche scientifique impulse et soutient le développement des activités relevant de son champ de compétence, et veille la mise en place des instruments de planification des activités relevant de son champ de compétence tous les échelons. A ce titre ;

- il propose les plans de développement de l'enseignement supérieur long, moyen et court terme,
- il anime, réalise ou fait réaliser toute étude prospective relative au développement des activités de l'enseignement supérieur,
- il veille au déploiement du réseau des établissements publics d'enseignement supérieur travers le territoire national conformément aux objectifs poursuivis par

le Gouvernement en matière d'aménagement du territoire et d'égalité d'accès aux cycles de l'enseignement supérieur,

- il oriente l'activité des établissements vers la satisfaction des besoins prioritaires du développement économique et social.
- il élabore et veille la mise en œuvre des plans d'équipements et matériels d'enseignement et de recherche scientifique des établissements d'enseignement supérieur,
- il veille l'application des dispositions légales et réglementaires relatives aux normes de sécurité, de travail et d'étude au sein des établissements d'enseignement supérieur,
- il définit les programmes d'investissements correspondants et en suit l'exécution,
- il élabore et veille l'application des mesures visant assurer une bonne maintenance des infrastructures, matériels et équipements,
- il assure la normalisation des installations et équipements des établissements d'enseignement supérieur en relation avec le système national de normalisation,
- il apporte, en matière d'intégration économique, son concours la promotion de la production nationale d'équipements, matériels ou produits d'utilisation courante dans les établissements de l'enseignement supérieur

Dans les domaines de la recherche scientifique et du développement technologique, le ministre de l'enseignement supérieur et de la recherche scientifique est compétent pour l'ensemble des activités et actions de recherche scientifique et de développement technologique, réalisées par les différentes structures. A ce titre, il est chargé notamment :

- En matière de recherche scientifique ;

- ✓ Proposer, élaborer, et mettre en œuvre la politique nationale en matière de recherche scientifique et de développement technologique,
- ✓ Proposer et mettre en œuvre les mesures permettant l'utilisation optimale des moyens nationaux de recherche scientifique et de développement technologique,

assurer la coordination des programmes de recherche fondamentale et appliquée des établissements d'enseignement supérieur,

- ✓ Veiller l'utilisation efficace des structures, équipements et autres moyens de recherche, soutenir les actions de vulgarisation de la science et de la technologie au sein de la société,
- ✓ Initier et faire aboutir, en concertation avec les autorités et instances concernées, toutes études relatives la définition des axes prioritaires de recherche, son intégration dans le développement économique, social et culturel du pays, et celles liées la localisation et l'implantation des structures de recherche,
- ✓ Veiller l'intégration des préoccupations de l'aménagement du territoire dans la politique nationale de recherche scientifique et de développement technologique,
- ✓ Préparer tous les éléments utiles aux travaux de planification, de programmation et de financement des activités de recherche scientifique et de développement technologique,
- ✓ Fixer, en liaison avec les secteurs et institutions concernés, les objectifs et les programmes nationaux de recherche scientifique et de développement technologique ainsi que les moyens concourant leur réalisation,
- ✓ Elaborer, proposer et assurer le suivi de la mise en œuvre des plans annuels et pluriannuels de recherche scientifique et du développement technologique correspondant aux programmes fixés, établir périodiquement les bilans relatifs l'état de réalisation des objectifs de la recherche scientifique et du développement technologique.

- En matière de développement technologique ;

- ✓ Organiser la veille technologique et suivre l'évolution des nouvelles technologies et de leur application dans les domaines économique, social et culturel,
- ✓ Fixer, en liaison avec les secteurs, institutions et opérateurs concernés, les objectifs et les programmes de développement technologique ainsi que les moyens concourant leur réalisation,
- ✓ Élaborer toutes études relatives aux conditions de mise en œuvre des projets et programmes de développement technologique,

- ✓ Mettre en œuvre les programmes de recherche scientifique et développement technologique dans les domaines fixés par la loi, mener toutes études ou travaux favorisant le développement de pôles technologiques dans le tissu industriel national.

Le ministre de l'enseignement supérieur et de la recherche scientifique assure le bon fonctionnement des structures centrales ainsi que des établissements publics placés sous sa tutelle. [18]

1.3 Quelques agrégats sur l'enseignement supérieur et la recherche scientifique (MESRS) :

1.3.1 Réseau des établissements universitaires :

Actuellement, l'université algérienne représente un réseau universitaire de **cent sept (107)** établissements d'enseignement couvrant quarante-huit (48) wilayas et répartis comme suit **:**

50	Universités dont celle de la formation continue
13	Centres universitaires.
31	Ecoles nationaux supérieur.
11	Ecoles normales supérieures
02	Annexes universitaires

Tableau 6 : Réseau des établissements universitaires.

1.3.2 Réseau des établissements de recherche :

Le réseau des établissements de la recherche sous la tutelle de l'année universitaire 2018/2019 qui était composé de quarante-cinq **(45)** établissements répartis comme suit :

07	Agences Nationales
12	Centres de recherches
26	Unités de recherche

Tableau 7 : Réseau des établissements de recherche.

1.3.3 Réseau des structures de soutien :

Il est composé de deux offices :

1- OPU : Office des Publications Universitaires.

2- ONOU : Office National des Œuvres universitaires.

1.3.4 Effectifs

a) Effectifs des étudiants : 1.730.000 étudiants.

b) Encadrement pédagogique : 63.948 enseignants répartis sur les différant grades (Professeur, Professeur hospitalo-universitaires, maître de conférences classe a et b,…).

c) Œuvres universitaires:

- Hébergés : 490 769 étudiants.

- Boursiers : 926 311 étudiants. [18]

1.4 Organigramme du MESRS :

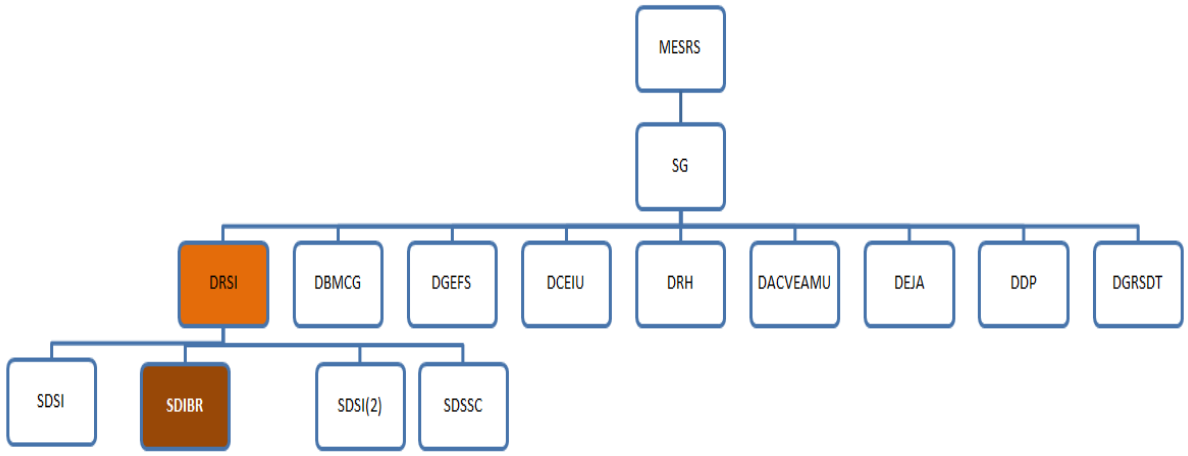


Figure 19 : Organigramme du MESRS.

Annexe : Présentation de l'organisme d'accueil

- MESRS : Ministère de l'Enseignement Supérieur et de la Recherche Scientifique.

-SG : Secrétariat Général.

-DRSI : Direction des Réseaux et Systèmes d'Information.

-DBMCG : Direction du Budget, des Moyens et du Contrôle de Gestion.

-DGEFS : Direction Générale des Enseignements et de la Formation Supérieur.

-DCEIU : Direction de la Coopération et des Echanges Inter=Universitaires.

-DRH : Direction des Ressources Humaines.

-DACVERAMU : Direction de l'Amélioration du Cadre de Vie des Etudiants et de l'Animation en Milieu Universitaire.

-DEJA : Direction des Etudes Juridiques et des Archives.

-DDP : Direction du Développement et de la Prospective.

-DGRSDT : Direction Générale de la Recherche Scientifique et du Développement Technologique.

-SDSI : Sous-Direction de la Sécurité Informatique.

-SDIBR : Sous-Direction de l'Infrastructure de Base et Réseaux.

-SDSI(2) : Sous-Direction des Systèmes d'Information.

-SDSSC : Sous-Direction des Systèmes de Support à la Connaissance.

La DRSICU comprend quatre (4) sous-directions :

a) **La sous-direction des infrastructures de base et réseaux (SDIRB)**, chargée :

- ✓ D'assurer l'intégration des infrastructures de base, des systèmes et des réseaux informatiques.
- ✓ De mettre en œvre la charte d'utilisation des ressources informatiques du secteur.
- ✓ De superviser les actions de maintenance et de gestion des systèmes informatiques du secteur.
- ✓ De mettre en place des références d'Elaboration des prescriptions techniques des réseaux locaux et des Equipements informatiques des Etablissements et veiller à leur mise en œuvre.
- ✓ De gérer la documentation dans son domaine de compétence. [18]

b) La sous-direction de la sécurité informatique(SDSI), chargée :

- ✓ De veiller à la sécurité informatique du secteur conformément aux règles en vigueur grace à des plans de sécurité physiques des sites.
- ✓ De mettre en place des mécanismes préventifs et curatifs pour le traitement des vulnérabilités, des alertes et attaques des réseaux et des systèmes informatiques du secteur.

- ✓ D'Evaluer périodiquement les besoins du secteur en matière de sécurité des systèmes informatiques, d'outils et de normes de sécurité informatique.
- ✓ D'assurer la protection des systèmes informatiques du secteur par la mise en place de mécanismes mutualisés de défense contre les virus et les programmes informatiques malveillants. [18]

c) La sous-direction des systèmes d'information (SDSI(2)), chargée :

- ✓ De veiller à la mise en œuvre d'un système collaboratif d'exploitation et de communication unifié du secteur.
- ✓ D'assurer le suivi des sites web des Etablissements du secteur pour une meilleure diffusion de l'information.
- ✓ De publier aux moyens des TIC, toute information relative au secteur ;
- ✓ De veiller à la mutualisation des droits d'utilisation des logiciels.
- ✓ De veiller à la promotion de la production des logiciels open source dans le respect Des droits d'auteur.
- ✓ De participer à la mise en œuvre de services en ligne dans le cadre de l'e-gouvernement.
- ✓ D'assurer la production et la promotion de services en ligne à destination des Etudiants, des enseignants-chercheurs, des chercheurs permanents, et des personnels du secteur.
- ✓ D'assurer la production de services en ligne à destination du citoyen ;
- ✓ De contribuer à l'Evolution du logiciel open source, en participant et en organisant des formations, colloques et séminaires.
- ✓ De faciliter la participation du secteur aux communautés de développeurs de logiciels open source.
- ✓ De gérer la documentation dans son domaine de compétence. [18]

d) La sous-direction des systèmes de support à la connaissance (SDSSC), chargée :

- ✓ De promouvoir l'utilisation des logiciels open source dans les cursus d'enseignement et de formation.
- ✓ De soutenir la création de contenus pédagogiques en soutien à la formation en présentiez dans le cadre d'une charte pédagogique nationale.
- ✓ De veiller au développement de la formation à distance.
- ✓ D'assurer l'informatisation des bibliothèques universitaires et leur interconnexion ; de mutualiser les ressources de production, d'Edition, de publication et de diffusion de contenus. [18]