SOLUTION HAUTE DISPONIBILITE AVEC REPLICATION DES DONNEES

Présentation de la solution

(d'après https://wapiti.telecom-lille.fr/commun/ens/peda/options/ST/RIO/pub/exposes/exposesrio2007/legrand-playez/index.html)

Le projet **Linux High Availability** a été développé dans le but de fournir une solution aux problèmes rencontrés précédemment. Cette solution répond à plusieurs impératifs : faible coût, facilité de maintenance et données parfaitement à jour en cas de bascule serveur. **Heartbeat** et **DRBD** sont les deux principaux modules issus de ce projet et les plus utilisés en environnement Linux dans la mise en place d'un cluster quand il s'agit de rendre des serveurs hautement disponibles.

Ces deux produits phares des solutions libres HA actuelles permettent de répondre à deux fortes contraintes : ne pas avoir de SPOF dans le cluster, et avoir des données parfaitement à jour en cas de bascule du service vers la deuxième machine. La solution consiste à mettre en place une solution de RAID-1 sur IP. C'est-à-dire un système de mirroring de partitions à travers une interface réseau.

Architecture cluster DRBD et Heartbeat

En temps normal, les services sont démarrés sur un seul nœud du cluster, appelé communément le nœud maître : il dispose de l'adresse IP sur laquelle est disponible le service, le système de fichier contenant les données est monté, et les différents services réseaux sont lancés. La deuxième machine (esclave) quant à elle, se contente d'attendre.

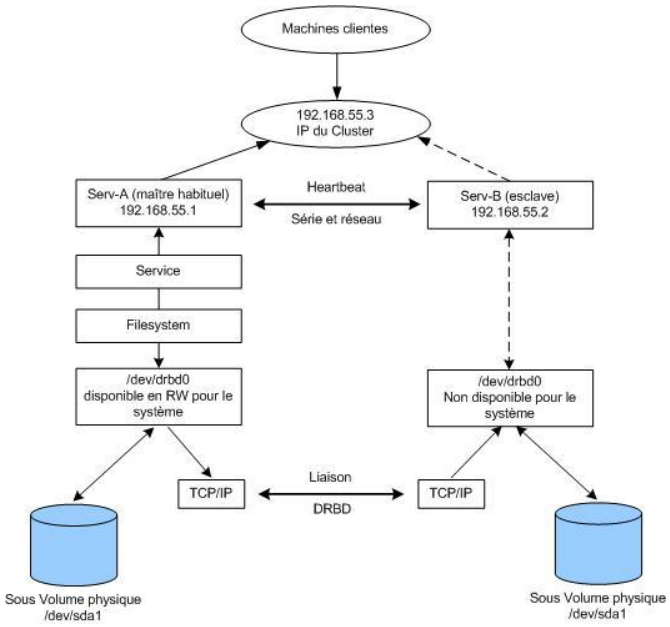
En cas de détection d'une panne sur le maître, les services sont alors basculés sur le nœud de secours, appelé communément le nœud esclave. Pour garantir la continuité apparente du service coté utilisateur, on utilise le principe des alias IP pour associer, en plus des adresses IP des membres du cluster, une adresse IP virtuelle à la machine qui héberge le service (maître).

Heartbeat utilise le programme **Fake** qui s'occupe de plus, de mettre à jour les tables ARP. Lors d'une bascule de cette adresse IP d'un membre à l'autre, Heartbeat met à jour les caches arp des éléments actifs du réseau par l'envoi de paquets arp gratuits (émission d'un « flood » de « gratuitous arp »).

Pour obtenir toutes les informations utiles sur l'état du cluster et décider éventuellement de basculer un service d'un nœud vers un autre, Heartbeat se base notamment sur un échange permanent de battements de coeur entre les deux machines. Le « pouls » des deux nœuds constituant le cluster est évalué par un système d'échange de messages. On utilise généralement plusieurs supports de communication pour fiabiliser cet échange. Une liaison Ethernet classique (pas forcément dédiée à Heartbeat) et un câble série reliant les deux nœuds du cluster sont souvent utilisés.

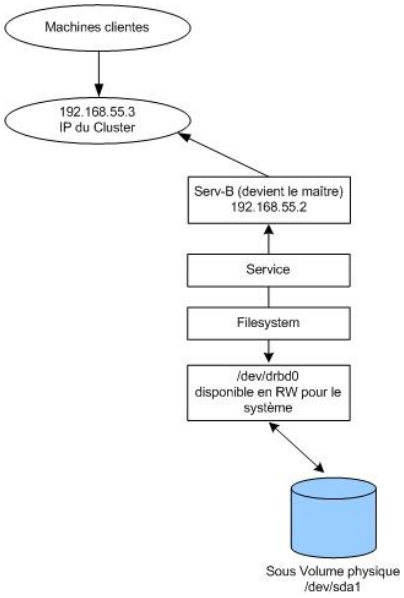
Les deux supports peuvent fonctionner simultanément afin d'éviter une fausse défaillance en cas de rupture d'un des liens. Chaque machine s'informe alors mutuellement du fonctionnement de l'autre par le biais d'Heartbeat.

Le schéma ci-dessous illustre l'architecture du cluster DRBD - Heartbeat en temps normal :



Dans la pratique, si le serveur maître présente un dysfonctionnement (en cas d'absence de signal pendant une durée déterminée sur les différents supports de communication), la machine esclave détecte l'arrêt des battements de cœur du maître par le biais de Heartbeat. Ce dernier lance alors une procédure de basculement totalement transparente pour l'utilisateur. Le serveur esclave acquiert alors l'adresse IP du service assuré par le cluster et effectue un ensemble d'actions (monter le système de fichier et réactiver les services par exemple) pour prendre le relais. Il est préférable d'utiliser le système de fichier journalisé ext3 afin de minimiser la vérification des données et donc d'assurer un basculement quasi instantané.

Voici le schéma du fonctionnement en mode dégradé (serveur A Hors Service) :

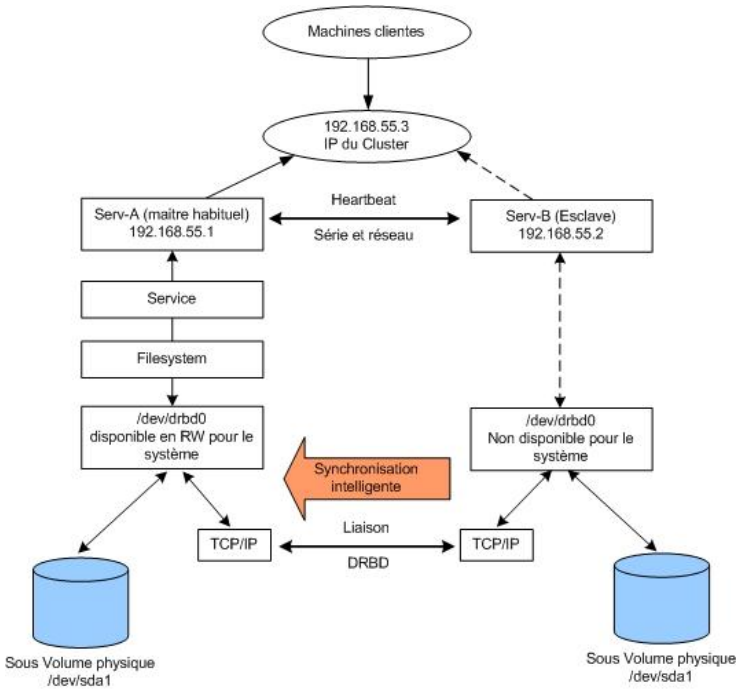


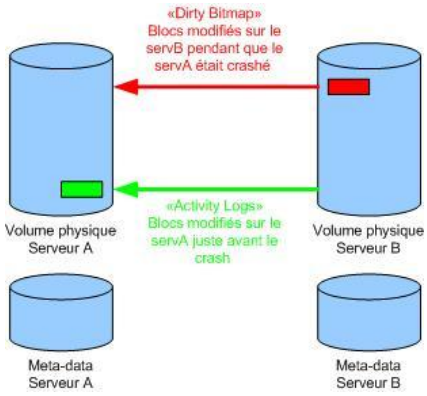
Lors du basculement et la reprise de service sur le serveur esclave après un crash du serveur maître, la partition répliquée est récupérée "en l'état" telle qu'elle était à l'instant du crash. On se trouve ainsi dans le même cas que lors de la reprise d'une partition locale après un crash. Une solution utilisant DRBD doit donc inclure des mécanismes de reprise de données, comme par exemple :

- La partition est utilisée par une base de données comprenant ses propres mécanismes de reconstruction des données (par exemple Oracle).
- Utilisation d'un système de fichiers classique avec vérification et réparation lors de la reprise (fsck). Cela peut s'avérer pénalisant en termes de performances et n'offre pas de garantie de réussite.
- Utilisation d'un système de fichiers journalisé assurant une reprise très rapide du système de fichiers et garantit qu'il est dans un état cohérent vis-à-vis du système (mais pas forcément des applications).

Dans les deux derniers cas, les applications gérant leurs données doivent elles aussi posséder un mécanisme de reprise de leurs données : par exemple, une application qui était en train d'écrire un fichier au moment du crash pourra retrouver un fichier tronqué lors de son démarrage sur le secondaire.

Lorsque le serveur en panne revient à l'état normal, il reprend son statut de maître, se re-synchronise avec l'actuel noeud maître. La bascule s'effectue alors sans interruption de service et en arrière plan.





La synchronisation des deux nœuds ne s'effectue que pour les blocs de données qui ont changé. On parle alors de synchronisation intelligente.