Exercie : NoSQL

Contexte : MMORPG

Localisation : Internationale, serveurs en Europe, Asie, Amériques, Océanie

* MySQL impossible à cause de la séparation des serveurs

Nous nous baserons sur le schéma CAP pour caractériser nos besoins et déterminer quel modèle de BDD nous utiliserons.

Les deux valeurs prioritaires sur le schéma sont : Availability et Partition tolerance (AP)

-Availability est prioritaire dans notre contexte, en effet la donnée doit être accessible en permanence, afin de garantir le service client à tout moment.

-Partition tolerance est sélectionné en raison de l’importance de la véracité des données, celle-ci garantissant la viabilité/jouabilité du jeu en lui-même.

-L’élimination du facteur Cohérence est justifié par la présence d’un index mis en place sur les serveurs, qui retournera la version la plus récentes (et donc exacte) des données requises quelque soit le serveur sur lequel elles sont stockées. Ceci entrainera probablement un délai supplémentaire lors de la récupération du à la consultation de l’index et à l’éventuel transfert des données depuis un serveur lointain et/ou stressé. De plus l’utilité d’infos concernant d’autres joueurs ou événements éloigné du joueurs (ex : l’inventaire d’un joueur ne peut être consulté que par lui même et pas par les autres joueurs, il n’y a donc pas d’impératif de cohérence entre les infos que les joueurs ont de l’inventaire des autres ) est faible et n’a donc pas un impératif de cohérence.

Le modèle de BDD sera orienté clef-valeur (exemple : SimpleBD, Memcached, Redis …)

Dans le cas d’une Database de MMO on ne peut se permettre de mettre en place un système sans mémoire tampon (la moindre action des joueurs émettraient une requête de changement se qui n’est pas gérable avec une communication directe joueur => DB , la mise en place de mémoire tampon (niveau serveur et niveau utilisateur) sera indispensable pour avoir des temps de réponse corrects. Cette mémoire tampon en local assurera la cohérence des infos du joueur pour lui-même.  
Availability peu être assuré par une mise en cache agressive des données importantes.

On se dirigera donc vers un sytème BASE pour la souplesse et la nécessité de modification avec des information qui seront en constante évolution et/ou incomplètes.