**A** - Utilisation du *setup* d’environnement par défaut du prof en déploiement.

**Problème 1** – Le MASTER de LOCUS ne semble pas reconnaître ses SLAVES puisqu’il produit tout le travail seul !

**Solution** : Réparer Locus avec nouvelle config de 10 WORKERS pour un MASTER (SLAVE non reconnu par cette version de LOCUS).

**B** - Utilisation de la majorité du *setup* d’environnement par défaut du prof, mais dont la partie LOCUS est ajustée pour reconnaître ses WORKERS et répartir l’envoi de requêtes.

**Problème 2** – Nouveau goulot d’étranglement, il s’agit soit du Webservice, soit de la base de données MySQL et probablement les deux !

**Hypothèse** : On croit que c’est les deux services qui posent un problème.

**Problème 2.1** - On va débuter par l’optimisation du Webservice, car il est plus facile à *scaler* et parce que c’est celui-ci qui consomme rapidement plus de 3 CPU.

**Hypothèse** : Un *scaling* horizontal du Webservice permettrait de répartir la charge entre les pods et ainsi empêcher une surcharge de ce service.

Pas scaling vertical parce qu’on lui a déjà attribué les ressources au besoin et monte toujours

Horizontal pod auto-scaling

Webservice horizontal pod autoscaling (gestion dynamique de la charge, s’agrafe au déploiement)

Si l’ajout de juste un pod à la fois, dans les moment de pic ou l’ajout de plusieurs, il va y avoir des problèmes de scaling. Pour cela, on va débuter à 5 pods de base.

1 pod pour commencer :

Dans webservice, on a mis une requests a 200m cpu, donc il va viser ça(80% du 200m cpu pour en créer un autre)

Hypothese avec un le pod va se faire écraser par plusieurs pods à un incrément de temps (erreurs), période de latence avec potentiels erreur/temps réponse augmenter

Probleme ne trouvait pas les metriques de cpu, solution utiliser la methode du prof et on voit en console que deploiement aurait acces aux metriques.

Hypothese vérifier donc un petit temps de latence qui perd quelques requête avant le nouveau dispatche (duplication), mais moins pire qu’estimé.

Hypothese : Solution trechold limite kill pod et le start tout de suite après (potentiels erreur avec requêtes qui disparaitres) pour repartir la charge mieux sur les autres pods

Résultats : start avec un pod est ok, amélioré la latence qui a diminué et on contrôle la consommation cpu

Probleme pods travaillent plus que d’autres avec une augmentation de la charge, exemple un pod prend bcp de chage alors que les autres sont la aussi avec moins de chage

On a debuté à 500mcpu, on a remarqué certains 200 et un a 1000

On vient d’identifier que mysql était a 800 mcpu donc un probleme potentiel debuté

Hypothse : essayer un treshold pour limiter chaque port et que kubernetes kill celui-ci a la limite avant d’en rouvrir un autre. La chage suplémentaire va se redistribuer aux autres, mais potentiel perte. Mais meilleur balancement et possibilite de revenir a un nombre accesptable de pods (horizontal auto-scaling)

Problématique 3500 erreurs, il ne seemble pas tuer le pod, c’est dure a voir. Courbe pour des requetes par seconde est vraiment en montagne russe. Au final, il prend plus de temps a créer les pods nécessaires au total. Les erreurs en plus (perdu) sont responsable de la fluctuation de charges et donc de l’ajustement des ressources.

Probablement un problème de restart, on doit setter quelque chose pour la config du restart

Après doc, on a changer le % a 50% de la limite, il devrait se rendre compte plus vite du load, moins de latence hypothese

Jusqua maintenant, le temps est constant parce qu’il ajuste très rapidement les pods, même trop vite potentiellement. Peut être qu’il agarder l’ancien hpa et predit l’augmentation de chage.

On a détruit et redeploye.

Il ne fait plus comme tantôt, donc il ne se souvenait pas du pic de chage et reagissait beaucoup moins bien. Pas de point de repere.

Solution potentiel avoir 5 pods depart, 5 pods qui offre un tampon pour se réaliser de l’augmentation de la chage.

On test le 5 pods. Limite de pord augmente a 800.

Resultat : la charge a pu se repartir sur plusieurs pods et il restait encore de la place encore du a la quantite de pods pour le stocage lors du scaling.

2000 users a 20 augmentation par seconde

On oscille entre 1000-2000 avec temps reponse 100-300ms, beaucoup oscillation.

On regarde donc nos log de webservice pour voir si maintenant le probleme ne serait pas au niveau de la bd. Puisque metric webservice belles, on pense pouvoir se diriger du coter mysql. De plus, les workers on l’air de bien repartir.