STACK LABS

Pascal Gillet <pascal.gillet@stack-labs.com>

[hippi-spark-k8s] Feedback - Suivi de mission

Pascal Gillet <pascal.gillet@stack-labs.com> 2 novembre 2020 à 15:48
À : Florence BRARD <FBRARD@sii.fr>, David Desviel <david.desviel@stack-labs.com>, Xavier Villalba <xavier.villalba@stack-labs.com>, Pascal ROUANET <PROUANET@sii.fr>, "QUINCHARD, Eric" <eric.quinchard@airbus.com>, "DELBENDE, Marc" <marc.delbende@airbus.com>

Bonjour à Tous,

Ce mail pour vous faire un état de l'avancée des travaux sur le projet hippi-spark-k8s.

Nous faisons un point de suivi hebdomadaire tous les lundis de 10h à 11h avec Éric Quinchard et Marc Delbende (nous avons donc déjà fait 2 points depuis le démarrage de la mission le 15/10).

Ces 2 premières semaines ont essentiellement été employées à reproduire et confirmer les conclusions de la pré-étude réalisée en interne chez Airbus DS quant au support de Kubernetes dans Spark.
Cette étape est nécessaire d'une part pour valider l'environnement de test (un cluster K8s dans Google Cloud) et d'autre part pour fixer les résultats attendus en termes d'intégration et d'exécution Python par rapport à une exécution scriptée.

Les tâches réalisées jusqu'ici:
- Création du cluster GKE.
- Submission d'un job via le Spark Operator (nécessite kubectl en local).
- Submission d'un job via spark-submit en mode cluster (nécessite une distribution de Spark en local).
- Submission d'un job via spark-submit en mode client en argument de l'image Docker (nécessite kubectl en local).
- Build d'une image Docker Spark (image officielle dans la distribution Spark) et utilisation dans tous les types de submission au-dessus.
- Tests du scheduling d'applications Spark avec Volcano et le scheduler par défaut dans K8s "kube-scheduler".
- Tests des nodes affinities: les drivers doivent s'exécuter dans des noeuds typés "driver" et les executors dans des noeuds "compute".
- Tests des priorités au queuing: s'assurer que les jobs mis en file d'attente sont bien triés en fonction de leur priorité ("scheduling order").
- Création d'une workload de test Spark "long-running-pi.py" qui dure suffisamment longtemps pour être représentatif des workloads HiPPI.
- Tests de préemption de jobs: OK quelque soit le scheduler (nécessite d'activer la préemption dans Volcano), mais pas supporté dans le Spark Operator car l'information de priorité n'est pas propagée dans les pods.

Suite à la réunion de ce matin:
- La submission d'un job via spark-submit en mode client en argument de l'image Docker sera la méthode privilégiée dans le Python.
- Cependant, la méthode via le Spark Operator n'est normalement pas plus "chère" à implémenter car elle repose sur les mêmes mécanismes (configuration YAML).
- Être capable d'identifier qu'un job a été préempté pour pouvoir le rescheduler ou le redémarrer, a minima le monitorer.
- Propager l'info de priorité jusque dans les executors.

Les premiers tests en Python vont être réalisés cette semaine.

Je laisse le soin à Éric ou à Marc de rebondir sur ce mail si besoin.

Cordialement,
--

STACK LABS
Pascal GILLET | Big Data & Cloud Architect
stack-labs.com
21 Boulevard de la Marquette, 31000 Toulouse
France · Canada