Prometheus remote-write

Contexte

Etude concernant le fonctionnement de remote-write et de remote-write receiver dans Prometheus.

Buts de l’étude :

- comprendre le fonctionnement de remote-write receiver
- comprendre le fonctionnement de remote-write
- remplacer la fédération par le mécanisme de remote-write

Caractéristiques remote write (1/2)

```
      |-->  queue (shard_1)   --> remote endpoint
WAL --|-->  queue (shard_...) --> remote endpoint
      |-->  queue (shard_n)   --> remote endpoint
```

- écriture des samples dans une queue appartenant à un shard à partir du WAL
- envoi sur le remote endpoint
- remplissage de la queue -> blocage de la lecture du WAL
- nombre optimal de shards calculé en permanence (fréquence de samples…)

Caractéristiques remote write (2/2)

Différentes options pour augmenter/réduire le débit :

```
queueConfig:
  minShards: 1
  maxShards: 200
  capacity: 2500
  maxSamplesPerSend: 500
  batchSendDeadline: 5s
  min_backoff: 30ms
  max_backoff: 5s
  [ retry_on_http_429: false ]
```

Architecture cible

```
                                    /-> Prom-0 destination(zone cloud)
Prometheus source(zone cliente) <
                                    \-> Prom-1 destination(zone cloud)
```

La connection est doublée pour ne pas perdre de métriques.

Configuration remote-write receiver (zone cloud)

Activable avec les options :

- `--enable-feature=remote-write-receiver` : expérimental dans notre version de Prometheus
- `--web.enable-remote-write-receiver` : stable dans les prochaines versions de Prometheus

Endpoint : `/api/v1/write`

Configuration remote-write (zone cliente) (1/2)

- Création d’un secret k8S contenant le token du user rancher

```
kubens caascad-monitoring
echo -n 'concourse-infra-pipelines-ocb-test06-cloud:xxxx' > ./tk.txt
kubectl create secret generic token-remote-write --from-file=./tk.txt
```

- Montage du secret dans le pod Prometheus

```
kubectl edit prometheus
secrets:
  - token-remote-write
```

Configuration remote-write (zone cliente) (2/2)

- Ajout d’une section RemoteWrite

```
remoteWrite:
  - bearerTokenFile: /etc/prometheus/secrets/token-remote-write/tk.txt
    name: prometheus-client-prometheus-0
    url: .../pods/...client-prometheus-0:9090/proxy/api/v1/write
    writeRelabelConfigs:
      - replacement: echo
        targetLabel: cc_prom_source
      - action: keep
        regex: ^(caascad)$
        sourceLabels:
        - caascad_com_prometheus_monitor_scope
      - action: labeldrop
        regex: caascad_com_prometheus_monitor_scope
```

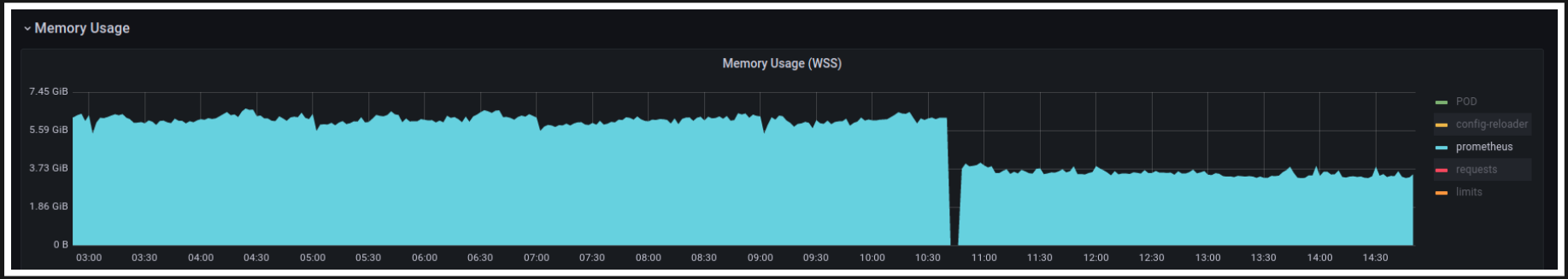
Envoi de métrique directement sur le remote-write receiver

Pour aider à débugguer, il est possible d’envoyer directement une métrique sur le receiver (remote write receiver, victoria-metrics…)

Contexte tests

- **Prometheus source** : version 2.33.5
- **Prometheus destination** (monitoring-client) : 2.32.1
- **Prometheus destination** (monitoring-app) : 2.33.5
- **Nombre de time series** : 353000

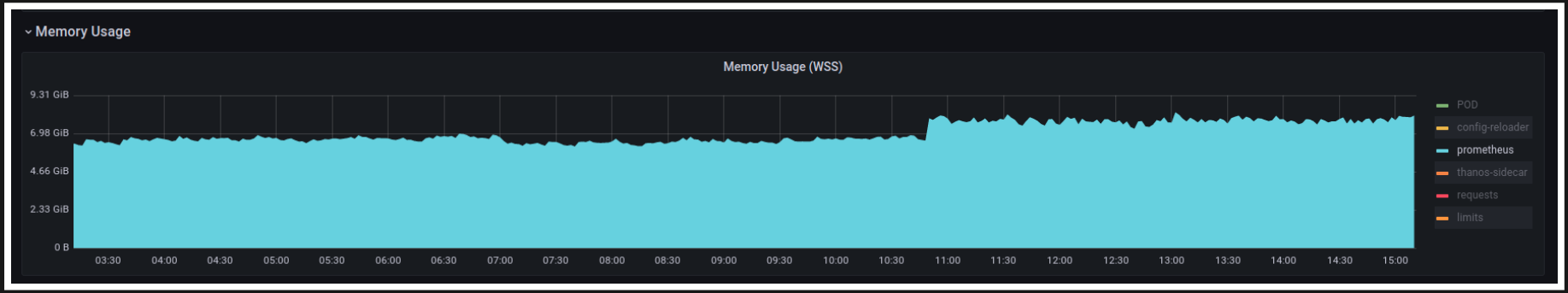
Différence mémoire cluster client



Fédération : =~ 3.50 GB

Remote write : =~ 6 GB

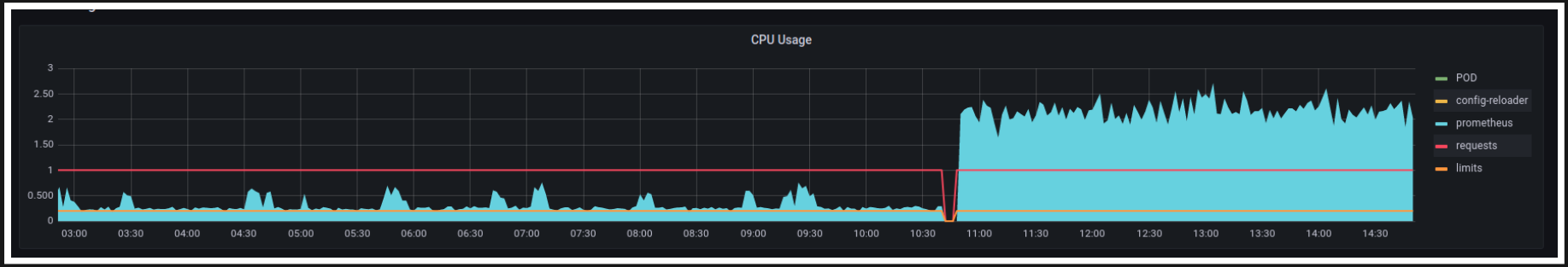
Différence mémoire cluster cloud



Fédération : =~ 7.80 GB

Remote write : =~ 6.90 GB

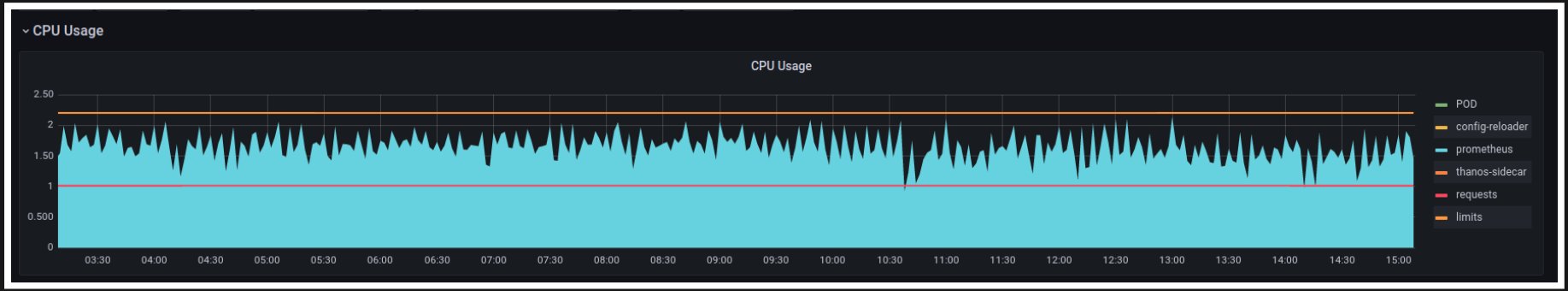
Différence CPU cluster client



Fédération : =~ 2.50

Remote write : =~ 0.5

Différence CPU cluster cloud



Fédération : =~ 2

Remote write : =~ 2

Tests de panne

Type d'erreur	Protocole de test	Résultat	Code HTTP
Authentification KO	Mauvais token côté conf cluster client pendant 10 minutes	Perte définitive de métriques pendant cette période	401 Unauthorized
Endoint Remote-write injoignable	Mauvaise url remote write côté conf cluster client pendant 10 minutes	Perte définitive de métriques pendant cette période	404 Not found
Remote-Write-Receiver KO 10min	Remote write receiver non disponible pendant 10 minutes	Pas de perte de métriques	503 Service Unavailable
Remote-Write-Receiver KO 1h15	Remote write receiver non disponible pendant 1h15 (WAL non chargé en block pendant cette période)	Pas de perte de métriques	503 Service Unavailable
Remote-Write-Receiver KO 1h	Remote write receiver non disponible pendant 1h (16h15 à 17h15) (WAL est flushé dans la TSDB 17h)	Pas de perte de métriques	503 Service Unavailable
Remote-Write-Receiver KO 2h20	Remote write receiver (prometheus monitoring-app) non disponible pendant 2h20	Perte de points pendant environ 1h20 : "Out of order sample from remote write"	400 Bad Request : out of bounds
VictoriaMetrics KO 2h20	Remote write receiver (victoria-metrics) non disponible pendant 2h20	Pas de perte de métriques	503 Service Unavailable

Métriques upstream (1/4)

Remote-write-receiver

Il n’y a pas de métriques spécifiques pour Remote-Write-Receiver. On peut seulement utiliser les métriques génériques habituelles (réseau et http principalement).

Exemple de métrique potentiellement intéressante:

`prometheus_http_requests_total{handler="/api/v1/write"}`
Cette métrique a un label `code` qui permet de savoir s’il y’a des requêtes en erreur.

Métriques upstream (2/4)

Remote-write

- `prometheus_remote_storage_samples_failed_t`
  nombre d’échecs d’envois sur le remote storage
  (**alerte upstream**)
- `prometheus_remote_storage_samples_retried_`
  : nombre de retry

Métriques upstream (3/4)

Remote-write

- `prometheus_remote_storage_shards_desired`: nombre de shards calculé selon le débit (**alerte upstream**)
- `prometheus_remote_storage_shards_max`: nombre de shards maximum (défini dans la conf) (**alerte upstream**)

Métriques upstream (4/4)

Remote-write

`prometheus_wal_watcher_current_segment` : segment courant du WAL en lecture

`prometheus_tsdb_wal_segment_current` : segment du WAL en écritures

`prometheus_remote_storage_highest_timestamp_in_seconds`: timestamp le plus elévé dans le WAL pour n'importe quel sample (**alerte upstream**)

`prometheus_remote_storage_queue_highest_sent_timestamp_` timestamp le plus elévé envoyé avec succès par remote write (**alerte upstream**)

Dashboards upstream

Deux dashboards disponibles :

- **Remote Write** : venant de la chart Helm kube-prometheus-stack
- **Remote Storage Stats**

Alertes upstream (1/2)

Remote-write-receiver

Pas d’alertes upstream.

Alertes upstream (2/2)

Remote-write

`PrometheusRemoteStorageFailures` : Prometheus éhoue à envoyer les samples au remote storage (severity critical, for 15 min, se déclenche notamment avec des erreurs 4xx)
`PrometheusRemoteWriteBehind` : Retard d'écriture (severity : critical, for : 15 min, se déclenche notamment avec des erreurs 5xx)
`PrometheusRemoteWriteDesiredShards`: Nombre de shards désirés est plus elevés que le nombre max configuré (severity : warning, for : 15 min)

Est-on suffisamment alerté ? (1/3)

Principaux problèmes :

- les métriques qui pourraient nous alerter sur un problème sur le remote write viennent elles-mêmes du remote-write
- les alertes upstream se déclenchent seulement au bout de 15 minutes

Est-on suffisamment alerté ? (2/3)

Proposition :

Remplacer l’alerte custom `FederatePrometheusDown` par une alerte spécifique pour le remote write

Est-on suffisamment alerté ? (3/3)

Pistes de solution :

Alerter quand la métrique `prometheus_http_requests_total` est absente ou quand `rate(prometheus_http_requests_total{handler="/api/v1/wr code!="204", namespace="monitoring-app"}[5m]) > seuil_a_determiner`

Les métriques remote write sont récupérées autrement que par le remote write

Conclusion (1/3)

Remote-write fonctionne bien et permet d'améliorer certains points :

- Différences de valeurs entre certaines recording rules et leurs expressions
- Tolérance temporaire aux pannes côté serveur (ce sont les erreurs 5xx), la panne ne doit pas être trop longue, sinon nous perdons quand même des métriques
- Problème de fédération : fréquence de scrapping trop elevée quand il y'a trop de données

Conclusion (2/3)

Remote-write fonctionne bien et permet d'améliorer certains points :

- Diminution usage CPU sur les clusters clients (2.5 → 0.5)
- Diminution usage mémoire sur le cluster cloud (7.80 → 6.90 GB)

Conclusion (3/3)

Par contre, certains poins négatifs persistent :

- Non tolérance aux longues pannes
- Non tolérance aux pannes côté client : perte définitive de métriques pendant cette période (ce sont les erreurs 4xx)
- Augmentation usage mémoire sur les clusters clients (3.50 → 6GB)

Si on remplace la fédération

- Upgrade Prometheus
- Création de nouveaux users Rancher / Suppression des anciens
- Revoir l'alerte `FederatePrometheusDown` pour la remplacer par une alerte spécifique remote write
- Revoir les R/L mémoire/cpu
- Fix dashboard/alerte upstream
- Et remplacement de la fédération pour les clusters où il n'y a plus de 1.13 🚀

Lien utile

- page confluence

Merci pour votre attention 😎