Spark and Flink running scalable in Kubernetes

Frank Conrad

Architect @ apomaya.com

scalable efficient low latency processing

frank@fc-tb.de, frank@apomaya.com

motivation, use case

- run (external, unknown trust) customer spark / flink code
- multi tenant cluster environment
- huge variation of job characteristics
- predictive job runtimes

Copyright © 2018 Apomaya

why kubernetes

- container, deployment, orchestration
- dynamic cluster
- support good CI/CD
  - help to try out
- help leverage scale of map-reduce pattern in cloud
- better cloud, provider, vendor agnostic
- leverage better cloud charge by used resources and time
- simpler version handling / migration

Copyright © 2018 Apomaya

jobs

- wide spread of needed memory / CPU / storage
- different SLA
- predictable job runtime
- different versions of spark / flink
- weekly, monthly,… processing, reprocessing, catchup

Copyright © 2018 Apomaya

multi tenant == isolation

- UI / API
- runtime
- network (CNI calico)
- storage
- logging / monitoring
- security

Copyright © 2018 Apomaya

the challenges of spark / flink with kubernetes

- cluster (manger) on top of cluster (manager)
- kubernetes drain, rescheduling,…
  - produce challenges to failure recovery of spark / flink
  - there are designed for single node failures but not rolling updates…
- kubernetes don’t like long running non restartable (statefull) apps
- multi tenant support is basic, a group is working on it (kubernetes-wg-multitenancy)
- immutable system images
  - updates need redeployment-> rescheduling

Copyright © 2018 Apomaya

a solution

- on demand deployment of spark/flink cluster, one per job
- job as first citizens
- each job get his right sized cluster
  - individual tuning possible
  - monthly, quarterly jobs, reprocessing, catchup have less impact to normal processing
- helper app
  - create / destroy spark / flink cluster (deployment)
  - submit, monitor job
  - report, integrate with higher level workflow (airflow,…)
  - optional proxy UI / API for tenant usage

Copyright © 2018 Apomaya

pros

- good separation of tenant
- real dynamic on demand spark/flink cluster
- allow wide tuning specific for job
  - CPU/memory / storage
  - SLA (use spot instances)
- easy run different version of spark/flink
- like “container“ for spark/flink jobs
- helper app can address kubernetes issues like drains (operator pattern)

Copyright © 2018 Apomaya

cons

- startup time of job depend on cloud provider
- overhead of more spark/flink infrastructure processes
- development of helper app, helm chart
- HDFS and shuffle

Copyright © 2018 Apomaya

spark 2.3 kubernetes integration

- the helper app: spark-submit, for many use cases
- deploy cluster, no helm need
- run driver
  - https://spark.apache.org/docs/2.3.0/running-on-kubernetes.html
  - https://github.com/apache-spark-on-k8s/spark

Copyright © 2018 Apomaya

helper app

- is the controller of the job and integrator to higher level workflow
- long running stream
  - suspend / resume support
- trigger other on demand resource
  - HDFS
- could based operator pattern
  - https://coreos.com/blog/introducing-operators.html
  - https://blog.couchbase.com/kubernetes-operators-game-changer/

Copyright © 2018 Apomaya

kubernetes helm

- helm chart
  - templating deployment
  - connect / deploy zookeeper,..
- can use all available kubernetes options
- local volumes
  - emptyDir (specify size)
- Persistent Volumes
  - local
  - EBS

Copyright © 2018 Apomaya

kubernetes auto scale

- cluster auto scaler
  - needed for real dynamic clusters
  - have proper labels / annotations for different instance types
  - you can have many different instance types for different need tries
  - make sure that auto scale group
    - really can scale to 0 instances, if you have the case
    - scale down with speed make sense

Copyright © 2018 Apomaya

kubernetes look to

- for predictive runtimes
  - set CPU/memory request == limit
- stability
  - OOM protection see above
  - volumes with size
  - JVM parameter must fit to limits (enough headroom for overhead / off heap)
- use persistent state
  - look to StatefullSet
  - stop and (re)start without disk data lost

Copyright © 2018 Apomaya

HA per job needed?

- really multi AZ needed?
  - pod failure can be handled by spark / flink
    - like single node failure
  - does not restart the hole job on AZ failure is ok?
    - on AZ failure anyhow 30-50% of job is lost
    - lower network latency, save network cost
- need zookeeper persistent or can ran only in memory?

Copyright © 2018 Apomaya

flink

- flink 1.5 make it simpler (FLIP-6)
  - HTTP/REST for all external communication
- S3 for checkpoint, snapshot,… see netflix : https://de.slideshare.net/FlinkForward/flink-forward-san-francisco-2018-steven-wu-scaling-flink-in-cloud
- Flink Forward Berlin 2017: Patrick Lucas - Flink in Containerland, https://de.slideshare.net/FlinkForward/flink-forward-berlin-2017-patrick-lucas-flink-in-containerland

Copyright © 2018 Apomaya

spark

- shuffle
  - external shuffle service?
- have sufficient local disk space for S3 based job output committer

Copyright © 2018 Apomaya

Logging / Monitoring

- Routing logs
  - fluentd (assign tags, routing,…)
- Monitoring / Metrics
  - Prometheus first choice at Kubernetes
    - it pulls (scrape) data from sources
    - push gateway for push
  - Grafana visualization / alternating
  - are health check sufficient?

Copyright © 2018 Apomaya

S3

- if performance problems
  - object name distribution: https://docs.aws.amazon.com/AmazonS3/latest/dev/request-rate-perf-considerations.html
- S3Guard: http://hadoop.apache.org/docs/r3.0.2/hadoop-aws/tools/hadoop-aws/s3guard.html
- https://de.slideshare.net/FlinkForward/flink-forward-san-francisco-2018-steven-wu-scaling-flink-in-cloud
- http://www.yonatanwilkof.net/spark-s3-parquet-aws-commiter-reliable-file-system-hdfs-hadoop-netfix/
- list of objects (data set is in the list and you select and filter on them)

Copyright © 2018 Apomaya

HDFS

- if S3 drive you crazy
  - does run your job only on small time per day
- HDFS on demand
- run it only if you need
  - stop if not used
  - then only billed by EBS but no EC2
- if don’t need persistent across restarts
  - could use local storage of nodes
- via a helper app, jobs can request that HDFS is get up

Copyright © 2018 Apomaya

stream app with external worker

- stream app run long time
- if you have large load difference over day time
  - auto scale cluster resources at runtime is a challenge
  - resources need to be configured for highest load
- if you offload the heavy resource part to an external micro service behind a load balancer
  - you can scale them up and down independent of the stream cluster
  - stream cluster must only configured for the maximum orchestration load

Copyright © 2018 Apomaya

Think different

Thank you

Questions?

Copyright © 2018 Apomaya