Notite licenta

Avem 2 variante:

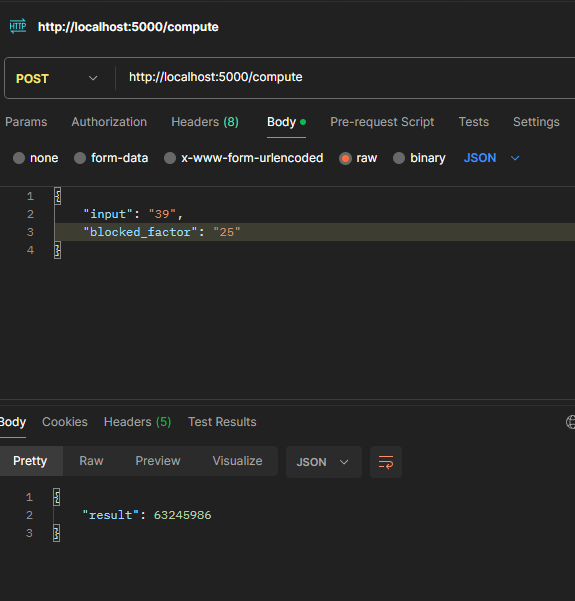
* Avand niste poduri predefinite, incercam sa asignam poduri random pe cluster si dupa sa antrenam agentul pe alte poduri deja puse pe cluster, astfel determinand un comportament al schedulerului in functie de imaginile de docker associate si alte taints ale manifesturilor
* Creem niste containere care blocheaza resurse si incercam sa antrenam pod-ul in conditii dinamice, generand astfel un comportament general al podului care nu depinde de celelalte containere deja puse pe cluster

Mergem pe a 2 a varianta pentru a facilita un comportament general si pentru a antrena modelul sa invete fiecare manifest cum s-ar comporta independent.

Avem 4 resurse pe care vrem sa le blocam:

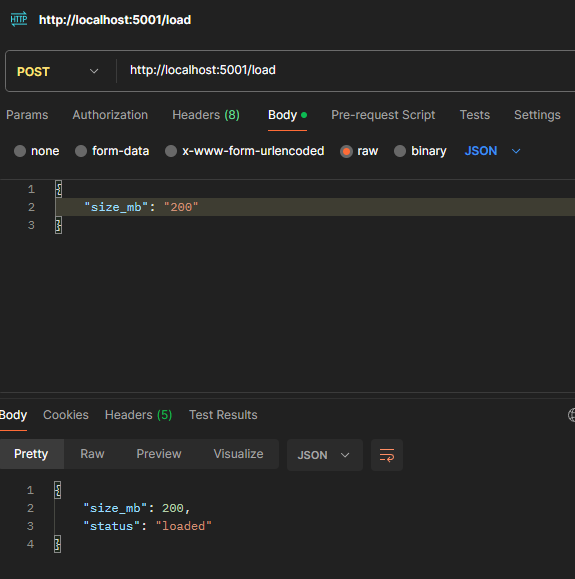
CPU – calculam Fibonacci si blocam core-uri

<http://localhost:5000/compute> POST - json cu input ( iteratii de fibonacii ) si blocked\_factor ( ce procent de cpu sa fie blocat aproximativ depinde de nr de core-uri )

 trebuie mai multe ajustari pe cluster, probleme cu docker stats sau cu docker container verificam pe cluster unde punem si monitorizari pe noduri

Am incercat sa blochez doar anumite core-uri pentru a putea simula un usage de 50% al procesorului dar metricile docker vad ca afiseaza tot 100%. Trebuie switch-uit la un cluster unde sa fie deployate imaginile alaturi de un container care sa genereze metrici de pe nod pentru a vedea mai in detaliu situatia

RAM – facem liste mari de numere float care sa ocupe spatiu determinat de noi, avand drept parametru spatiul pe care noi vrem sa il ocupam

 exista mici variatii fata de ce e ca parametru

--- acestea au fost facute pe 28.02

DISK – scrierea pe disk care s-ar face cu scrierea de numere intr-un fisier si citirea acestora in 2 thread-uri diferite din 2 fisiere diferite – lucram cu docker volume cel mai probabil

* O sa avem 3 moduri – write, read si readwrite care sa asigure ca scheduler-ul este atent la read si write si aloca pod-uri bazandu-se pe necesitatile acestuia

Am facut componenta de disk blockage dar docker metrics nu vede scrierea si citirea din fisier. De asemenea, pentru a nu consuma tot hdd-ul am facut o citire repetitiva peste un fisier de 100mb si scrierea si stergerea repetata a unui fisier de 10mb. De asemenea, pare sa blocheze cpu-ul, cred ca voi scrie aceleasi lucruri statice pentru a nu consuma cpu generand numere random

Bandwidth – trimiterea de pachete mari intr-un request spre undeva sa fac ddos ( ma gandesc la o varianta spre router )

Am scris componenta de network blockage care genereaza date random si face un stream cu ele pe un request de https catre container

TO DO:

* Terminat containere de mockuit resurse
* De facut cluster si de verificat functionalitate si de adaugat solutii de metrici
* De facut pod uri de testat against ( server python fullstack pt cpu ram si bandwidth de generat imagini, etc)
* de stabilit algoritmul de rl si de schitat environment cu metricile obtinute
* de legat environment rl de cluster prin requesturi
* de scris scheduler si de adaugat la k8 bazandu-se pe rezultatul rl
* de comparat rezultate

am creat scripturi pentru containerizarea rapida a blockerelor de resurse si pentru pushuirea acestora intr-un registry local pentru a nu le buildui de fiecare data ( economisim timp si resurse ) si am marit capacitatea wsl2 pentru a putea sustine cluster-ul de tip kind cu 3 noduri.

Initial voiam kind dar nu expune metrici low level ca un nod cu un intreg linux os

O sa folosim mai multe kernel-based virtual machine (KVM) pentru a simula niste noduri cu linux care sa ne dea si metrici low leve si kubdeadm.

In fiecare nod kvm o sa avem linux care sa ruleze pod-uri asignate de kubernetes master node. Docker registry cred ca trebuie mutat in docker hub pentru a putea permite kvm-ului sa ajunga la el.

Ca os pentru kvm-uri avem imaginea de linux folosita de cloud provideri pentru diversi agenti ( lightweight menita sa ruleze si pe bare metal).

Avem cloud init pentru configurarea serviciilor generale pentru kvm

Unul pentru master node si unul pentru worker nodes

in cadrul acestor cloud init config files avem configurari pentru os, pachete de instalat pe acesta si comenzi de rulat in acesta. Astfel, printr-o comanda de initializare a kvm-ului cu acest config file, se pot rula automat toate comenzile de pornire si configurare a cluster-ului.

Vor fi un master node si 3 worker nodes, fiecare cu 2gb ram si 2 core-uri. Pentru a facilita resursele pentru kvm uri, in cadrul configurarilor windows am marit resursele alocate catre wsl2, astfel acesta avand in total 10 core uri si 10gb ram.

Ma bat cu greu cu kvm-uri, probleme la configurare cu cloud init – initiativa canonical, metoda prin care sunt creeate toate vm-urile din cloud. Am luat log-urile din creearea vm-ului pe masina locala pentru a verifica pasii si pentru a vedea erorile

Am reusit sa configurez kvm-urile pe o interfata de internet creata odata cu ele, am instalat pachetele de administrare kubernetes si serviciul de rulare a containerelor Docker. Toate configurarile se realizeaza din script-uri de start, join, deleteContext si stop.

Trecem pe linux. Pe master node containerele de kube system fail-uiesc destul de random, semn ca nu ar avea destule resurse. Mutam pe linux pentru a putea asigura mai multe core-uri si mai mult ram.

Am trecut env de dev pe linux nativ pe laptop pt mai multe resurse libere. Avem o comunicare intre noduri, merg cum trebuie kvm-urile doar ca containerele de kubeadm pica din motive random. Am găsit ca kubeapi-server crapa din cauza etcd din loguri

W0328 22:19:17.035054 1 logging.go:55] [core] [Channel #22 SubChannel #23]grpc: addrConn.createTransport failed to connect to {Addr: "127.0.0.1:2379", ServerName: "127.0.0.1:2379", }. Err: connection error: desc = "transport: Error while dialing: dial tcp 127.0.0.1:2379: connect: connection refused"

și în etcd am găsit asta în loguri:

{"level":"info","ts":"2025-03-28T22:19:00.900839Z","caller":"osutil/interrupt\_unix.go:64","msg":"received signal; shutting down","signal":"terminated"}

Deci cineva îmi omoară etcd-ul dar cine

gata l am găsit

Mar 28 22:41:05 k8s-node-master kubelet[3234]: E0328 22:41:05.946229 3234 event.go:359] "Server rejected event (will not retry!)" err="Timeout: request did not complete within requested timeout - context deadline exceeded" event="&Event{ObjectMeta:{etcd-k8s-node-master.18311780621bdbaa kube-system 1281 0 0001-01-01 00:00:00 +0000 UTC <nil> <nil> map[] map[] [] [] []},InvolvedObject:ObjectReference{Kind:Pod,Namespace:kube-system,Name:etcd-k8s-node-master,UID:f05bda3a2f28b4df60c28cfabc1b1940,APIVersion:v1,ResourceVersion:,FieldPath:spec.containers{etcd},},Reason:Unhealthy,Message:Readiness probe failed: Get \"http://127.0.0.1:2381/readyz\": dial tcp 127.0.0.1:2381: connect: connection refused,Source:EventSource{Component:kubelet,Host:k8s-node-master,},FirstTimestamp:2025-03-28 22:19:30 +0000 UTC,LastTimestamp:2025-03-28 22:40:31.943344257 +0000 UTC m=+1291.217463830,Count:5,Type:Warning,EventTime:0001-01-01 00:00:00 +0000 UTC,Series:nil,Action:,Related:nil,ReportingController:kubelet,ReportingInstance:k8s-node-master,}"

asta am găsit în log-urile os-ului de pe master node. Deci pica proba de readiness și de asta se tot restarteaza etcd-ul și nu merge nimic

în logurile de la kubelet am găsit aceasta linie:  
Mar 31 14:05:55 k8s-node-master kubelet[3246]: E0331 14:05:55.924835 3246 event.go:368] "Unable to write event (may retry after sleeping)" err="Post \"https://192.168.122.136:6443/api/v1/namespaces/kube-system/events\": dial tcp 192.168.122.136:6443: connect: connection refused" event="&Event{ObjectMeta:{etcd-k8s-node-master.1831e840c85f4d82 kube-system 0 0001-01-01 00:00:00 +0000 UTC <nil> <nil> map[] map[] [] [] []},InvolvedObject:ObjectReference{Kind:Pod,Namespace:kube-system,Name:etcd-k8s-node-master,UID:0be9da42a461eb89b3f7e0e7f21d6ab0,APIVersion:v1,ResourceVersion:,FieldPath:spec.containers{etcd},},Reason:Killing,Message:Stopping container etcd,Source:EventSource{Component:kubelet,Host:k8s-node-master,},FirstTimestamp:2025-03-31 14:04:55.706332546 +0000 UTC m=+0.179260185,LastTimestamp:2025-03-31 14:04:55.706332546 +0000 UTC m=+0.179260185,Count:1,Type:Normal,EventTime:0001-01-01 00:00:00 +0000 UTC,Series:nil,Action:,Related:nil,ReportingController:kubelet,ReportingInstance:k8s-node-master,}"

în care practic încearcă să se noteze faptul ca s-a trimis semnalul de kill către etcd însa faptul ca etcd e mort nu lasa api-ului sa noteze și e o chestie circulara.

Am rezolvat cu etcd-ul, adaugand un delay în comenzile de startup din cloud init config sa aștept pana este ready prin endpoint. Celelalte containere continua sa faileze și acum sa vedem de ce.

Sunt probleme la PLEG: pod lifecycle event generator. Practic nu mi se sync-uiesc pod-urile sa poată fi urmărite de containerd și e puțin haos

O luam de la capăt cu creearea cluster-ului și încerc sa fac de mana manual un cluster și după încerc sa automatizez.

Pt KVM-uri trebuie dezactivat swap-ul pentru k8s.

Am renunțat la cloud config pentru a rula comenzile automat, încerc sa fac un plan de bătaie pentru script-uri și pentru fisierele modificate sa le incarc de mana în kvm-uri și sa fac script-uri care sa ruleze cu sudo în cadrul acestora.

sudo apt update

sudo apt install docker.io -y

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt update

sudo apt install kubeadm kubelet kubectl -y

sudo apt-mark hold kubeadm kubelet kubectl

**# avem kubernetes pe masini - comenzile de mai sus trb pe toate vm-urile**

**# trb sa aducem containerd.conf pe toate vm-urile - il lasam la ~**

**# script de rulat pe masina**

scp utilFiles/containerd.conf kubernetes@IP:~

scp utilFiles/kubernetes.conf kubernetes@IP:~

scp utilFiles/kubelet kubernetes@IP:~

**# la fel cr ca merge si pt niste script uri de rulat in vm**

scp utilFiles/scriptMaster

scp utilFiles/scriptWorker

**# asta pt fiecare ip**

**# inapoi pe fiecare kvm**

cp ~/containerd.conf /etc/modules-load.d/containerd.conf

sudo modprobe overlay

sudo modprobe br\_netfilter

cp ~/kubernetes.conf /etc/sysctl.d/kubernetes.conf

sudo sysctl --system

am început rescrierea script-ului de creeare a mașinilor virtuale efectuand configurarea de kubernetes și alte configurari din script-uri copiate și rulate prin ssh

am rescris script-urile de creeare a kvm-urilor, avem containere stabile de kubeadm pe master node, trebuie reparat node network manager-ul, în cazul nostru flannel

gata avem script automat de creeare a unui cluster yupy, trb puțin optimizat ca mănâncă multe resurse acum

Având cluster-ul gata, acum revenim la creearea aplicatiilor de blocare a resurselor.

O să rescriu aplicatiile din 4 aplicații separate intr un singur rest api cu flask care, în funcție de call-uri, blochează anumite resurse, și astfel prin deploy-erea unui singur pod cu aceasta aplicație pe fiecare nod din cluster, pot altera resursele disponibile de pe fiecare nod. Eliminam și overhead-ul necesar sa gestionam 4 pod-uri cu 4 aplicații separate care fiecare sa blocheze o singura resursa.

Am modificat aplicația și avem un singur server de python care primește request-uri care blochează anumite resurse. Am făcut și documentatie cu swagger pentru o testare mai ușoară

Voi avea 2 functionalitati.

Întâi vom avea antrenare care include și numele podului – modelul învața comportamentul container-ului pe care îl deployaza. Aici vom avea 2 categorii:

- la nivel de aplicație întreaga unde modelul alege după numele aplicației și știe toate componentele și cum ar interactiona intre ele

Aici, practic vrem un scheduler la nivel de proiect / aplicație care sa înțeleagă comportamentul intre componente și servicii și sa poată vedea când face scheduling fiecare nod ce pod-uri are, ce pod are de deployat și astfel sa ia alegerea

- la nivel de pod unde se învața comportamentul unui singur pod

Aici, scheduler-ul învața comportamentul individual al unui pod și astfel vede la fiecare nod ce resurse are și este fine tuned pt un anume pod

Acum încep sa construiesc aplicația, care va avea 4 componente:

- frontend : react – va folosi mainly network bandwidth și niște cpu

- backend : c# - va folosi mainly cpu și niște ram

- caching: redis – va folosi mainly ram

- db: ? Încă nu stiu – va folosi mainly i/o