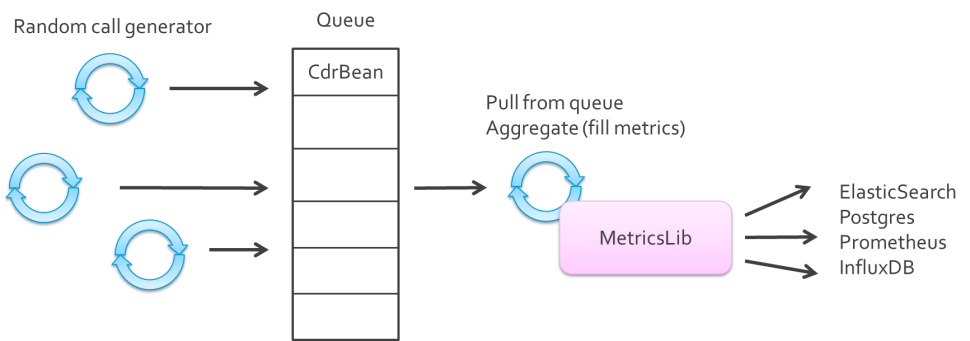
**PMON – Simulator klicev**

Navodila za uporabo

Matjaž Cerkvenik, Iskratel 2020

**Opis**

Simulator je bil razvit z namenom testiranja Iskratelove rešitve za performance monitoring – PMON. Hkrati je namenjen testiranju MetricsLib knjižnjice, ki je del PMON rešitve in je odgovorna za vpisovanje metrik v ElasticSearch, ter prilaganjanje podatkovnega modela metrik v time-series podatke in učinkovito uporabo indeksov v Elasticsearchu.



Simulator na eni strani sestoji iz množice niti, ki v enakomernih časovnih presledkih generirajo namišljene klice – tj. klice z naključno izbranimi vrednostmi: A-številka, B-številka, trajanje klica, release cause, vhodne/izhodne trunk grupe ipd.

Na drugi strani se iz seznama zgeneriranih klicev podatki pretvorijo v time-series podatkovno strukturo (metrike, agregacije). MetricsLib knjižnjica je vgrajena v simulator in omogoča pošiljanje metrik na ElasticSearch. ElasticSearch klient deluje kot eno-niten process.

Generirane metrike so zelo dober približek realnemu stanju v produkciji.

Docker image se nahaja na Docker Hub-u:

<https://hub.docker.com/r/matjaz99/cdrpr>

Izvorna koda simulatorja se nahaja na Github-u:

<https://github.com/matjaz99/cdrpr>

Compose datoteke se nahajajo v direktoriju deployment.

**Zagon simulatorja**

Simulator deluje kot kontejner v Docker okolju.

Compose datoteka:

version: '3.6'

networks:

net:

driver: overlay

attachable: true

services:

cdrsim:

image: matjaz99/cdrpr:2.0

networks:

- net

ports:

- 9099:9099

environment:

- CDRPR\_THREADS=32

- CDRPR\_SIMULATOR\_DELAY=30

- CDRPR\_SEND\_INTERVAL\_SEC=60

- CDRPR\_ES\_HOST=172.29.6.6

- CDRPR\_ES\_PORT=9200

- CDRPR\_ES\_AUTO\_CREATE\_INDEX=false

- CDRPR\_DUMP\_TO\_FILE=false

deploy:

mode: replicated

replicas: 1

Zagon ElasticSearch in simulatorja v Docker swarm okolju:

$ docker stack deploy -c compose-es.yml es

$ docker stack deploy -c compose-cdrsim.yml cdrsim

**Konfiguracija**

Z ustrezno konfiguracijo je mogoče kontrolirati količino zgeneriranih klicev in posledično količino zbranih metrik. Simulator je mogoče nastavljati preko okoljskih spremenljivk, ki jih nastavimo na kontejnerju v docker okolju (glej compose datoteko).

Na količino generiranih metrik vplivata parametra: CDRPR\_THREADS in CDRPR\_SIMULATOR\_DELAY. Spodnja tabela prikazuje približno količino generiranih metrik:

|  |  |  |
| --- | --- | --- |
| CDRPR\_THREADS  Število niti, ki generirajo klice. | CDRPR\_SIMULATOR\_DELAY  Interval kako pogosto vsaka nit generira klice | Število generiranih metrik |
| 8 | 20 | 40k |
| 16 | 20 | 50k |
| 32 | 20 | 70k |
| 64 | 20 | 110k |
| 16 | 10 | 80k |
| 32 | 10 | 110k |
| 64 | 10 | 160k |

S parametrom CDRPR\_SEND\_INTERVAL\_SEC se nastavlja kako pogosto se metrike pošiljajo v ElasticSearch. V produkciji, se bodo metrike pošiljale na 15 min. Za namene testiranja je mogoče periodo zmanjšati npr. na 1 min.

Vse metrike, ki pripadajo istemu indeksu se v ElasticSearch vpišejo z eno REST zahtevo (bulk insert):

POST /\_bulk

{ "index":{ "\_index":"pmon\_cdr\_calls\_idx"} }

{"m\_name":"pmon\_cdr\_calls\_by\_cause","node":"Prague","cause":"Other","trafficType":"OUT","value":520.0,"timestamp":1600252760361}

{ "index":{ "\_index":"pmon\_cdr\_calls\_idx"} }

{"m\_name":"pmon\_cdr\_calls\_by\_cause","node":"Ljubljana","cause":"Other","trafficType":"TRANSIT","value":503.0,"timestamp":1600252760361}

…

Če imamo 10 indeksov, to pomeni 10 POST zahtev. Podatki se v ElasticSearch vpišejo v roku nekaj sekund, kar ne predstavlja performančnega problema. Glej tabelo metrik.

S parametroma CDRPR\_ES\_HOST in CDRPR\_ES\_PORT se nastavi hostname ali IP naslov ElasticSearch strežnika kamor se pošiljajo metrike in port kjer posluša ElasticSearch.

V primeru, da je vpisovanje v ElasticSearch neuspešno, MeticsLib poizkuša 3x ponovno vpisati. Nastavljivo s parametrom: CDRPR\_RETRIES; na koncu metriko zavrne.

MetricsLib omogoča avtomatično kreiranje indeksa in mappinga, če le-ta še ne obstaja. Funkcionalnost je mogoče vključiti s CDRPR\_ES\_AUTO\_CREATE\_INDEX=true. Privzeto je funkcionalnost izključena in uporabnik si mora sam ročno nastaviti ustrezen mapping.

Zato, da se ElasticSearch zaveda časa v metrikah, je obvezno (ročno ali avtomatično) nastaviti vsaj mapping za polje timestamp (unix time in millis).

**Index lifecycle policy**

Pred prvim zagonom simulatorja je potrebno v ElasticSearch skonfigurirati politiko upravljanja z indeksi. To je potrebno izvesti ročno pred prvim zagonom simulatorja. Za ta namen je pripravljena posebna shell skripta, ki se jo zažene z ukazom:

$ sh configure\_elastic.sh <hostname>

Namesto <hostname> je potrebno vpisati IP naslov ali hostname ElasticSearch strežnika kamor se pošlje konfiguracija.

Konfiguracija izvede korake:

1. Nastavitev Elastic gruče:

PUT /\_cluster/settings

{

"persistent": {

"indices.lifecycle.poll\_interval": "10m",

"slm.retention\_schedule": "\* \* \* \* \* ?"

}

}

1. Nastavitev politike upravljanja indeksov (za rotiranje indeksov):

PUT /\_ilm/policy/pmon\_ilm\_policy

{

"policy" : {

"phases" : {

"hot" : {

"actions" : {

"rollover" : {

"max\_size" : "500mb",

"max\_docs" : 100000000,

"max\_age":"3h"

},

"set\_priority": {

"priority": 50

}

}

},

"warm" : {

"min\_age" : "120m",

"actions" : {

"readonly" : { },

"allocate" : {

"number\_of\_replicas" : 0

},

"forcemerge": {

"max\_num\_segments": 1

},

"shrink": {

"number\_of\_shards": 1

},

"set\_priority": {

"priority": 25

}

}

},

"cold" : {

"min\_age" : "6h",

"actions" : {

"allocate" : {

"number\_of\_replicas" : 0

},

"freeze" : { },

"set\_priority": {

"priority": 0

}

}

},

"delete" : {

"min\_age" : "3d",

"actions" : {

"delete" : { }

}

}

}

}

}

**Kreiranje indexov**

Simulator bo pred začetkom vpisovanja preveril ali index oz. njegov alias obstaja. Če obstaja, potem simulator začne vpisovati podatke.

Če index še ne obstaja, bo simulator najprej kreiral mapping, ki ga bo povezal s politiko upravljanja indeksov (pmon\_ilm\_policy):

PUT /\_template/pmon\_cdr\_trunk\_calls\_idx\_tmpl

{

"order": 10,

"index\_patterns": ["pmon\_cdr\_trunk\_calls\_\*"],

"settings": {

"index": {

"number\_of\_shards" : 1,

"number\_of\_replicas" : 0,

"lifecycle.name": "pmon\_ilm\_policy",

"lifecycle.rollover\_alias": "pmon\_cdr\_trunk\_calls\_idx"

}

},

"mappings": {

"properties": {

"metric\_name": {"type": "keyword"},

"value": {"type": "double"},

"timestamp": {"type": "date", "format": "epoch\_millis"}

}

}

}

Nato bo simulator kreiral še indeks s pripadajočim aliasom:

PUT /pmon\_cdr\_trunk\_calls\_idx-000000

{

"aliases": {

"pmon\_cdr\_trunk\_calls\_idx":{

"is\_write\_index": true

}

}

}

**Metrike**

Z metrikami (time-series model) štejemo število telefonskih klicev glede na dane labele v metriki. Tipično nas zanima število klicev na določenem omrežnem elementu, ter vhodni in izhodni prenosniki (trunki), ali pa število odgovorjenih/zasedenih/neodgovorjenih klicev.

Podatkovni model metrik:

{

"metric\_name": "pmon\_cdr\_calls\_by\_cause\_count",

"nodeId": "Node1",

"cause": "Answered",

"incTgId": "1000",

"timestamp": 15324568722,

"value": 12345

}

*Metrika sporoča, da je bilo na omrežnem elementu ‘Node1’, na vhodni trunk grupi ‘1000’, 12345 odgovorjenih (‘Answered’) klicev.*

Vsaka metrika sestoji iz imena metrike, iz niza poljubnih label (key=value), timestamp (UNIX time v milisekundah) in sama vrednost metrike. Vsa polja so tipa 'String', razen timestamp je tipa 'Long' in value je tipa 'Double'.

Ime metrike, timestamp in vrednost so obvezni parametri vsake metrike, ostale labele pa so poljubne in se lahko tudi spremenijo (sicer redko, ampak možno jih je prilagajati potrebam).

Glede na kontekst metrike je smiselno, da se različne metrike zapisujejo v različne indekse v ElasticSearch.

V primeru, da je avtomatično kreiranje indeksa omogočeno, se za vsak indeks ustvari tudi ustrezen mapping:

PUT /<index>

{

"mappings": {

"properties": {

"metric\_name": {"type": "keyword"},

"value": {"type": "double"},

"timestamp": {"type": "date", "format": "epoch\_millis"}

}

}

}

*Opomba: indeks vsebuje mapping le obvezne parametre: metric\_name, value in timestamp.*

Simulator generira naslednje metrike:

|  |  |  |  |
| --- | --- | --- | --- |
| **Index** | **Metrika** | **Labele** | **Ocenjeno število metrik (max)** |
| pmon\_cdr\_calls\_idx | pmon\_cdr\_calls\_by\_cause\_count | node  cause  trafficType | 3.000 |
| pmon\_cdr\_calls\_in\_progress | node | 100 |
| pmon\_cdr\_call\_durations\_idx | pmon\_cdr\_calls\_by\_duration | node | 100 |
| pmon\_cdr\_call\_timings\_idx | pmon\_cdr\_time\_before\_ringing | node | 100 |
| pmon\_cdr\_time\_before\_answer | node | 100 |
| pmon\_cdr\_trunk\_calls\_idx | pmon\_cdr\_trunk\_calls | node  cause  incTG  outTG | 100.000 |
| pmon\_cdr\_trunk\_durations\_idx | pmon\_cdr\_trunk\_calls\_duration | node  incTG  outTG | 10.000 |
| pmon\_cdr\_business\_group\_idx | pmon\_cdr\_bg\_calls | node  bgidOrig  bgidTerm | 1.000 |
| pmon\_cdr\_cg\_calls | node  cgidOrig  cgidTerm  centrexCallType  ctxCall | 3.000 |
| pmon\_cdr\_supplementary\_service\_idx | pmon\_cdr\_supplementary\_service | node  servId  servIdOrig  servIdTerm | 1.000 |
| pmon\_cdr\_subscriber\_group\_idx | pmon\_cdr\_subscriber\_group\_calls | node  callingSubscrGroup  calledSubscrGroup | 1.000 |
| pmon\_cdr\_voip\_idx | pmon\_voip\_rx\_codec | codec | 100 |
| pmon\_voip\_tx\_codec | codec | 100 |
| **SKUPAJ (groba ocena):** |  |  | **~120.000 metrik** |

**Endpoints**

Privzeto MetricsLib deluje na portu 9099 in izpostavlja naslednje URL endpointe.

Konfiguracija MetricsLib-a ter vse registrirane metrike (vsak registry predstavlja svoj indeks v Elasticu):

<http://localhost:9099/hello>

Metrike v Prometheus formatu:

<http://localhost:9099/metrics>

Hitri pregled indeksov v Elasticsearchu:

<http://localhost:9099/indices>

**Logiranje**

Simulator izpisuje log zapise v standard output (v terminal). V kontejnerju si lahko log zapise ogledamo s komando:

$ docker service logs -f cdrsim\_cdrsim

V logih je videti zapisovanje v ElasticSearch v naslednji sintaksi:

>>> sending: <ime\_metrike> [size=<število\_metrik>]

<<< POST <url\_endpoint> - <http\_error\_code>

Primer:

INFO: EsClient[0]: >>> sending: metric pmon\_cdr\_calls\_by\_cause [size=635]

INFO: EsClient[0]: <<< POST http://elasticvm:9200/\_bulk - 200

INFO: EsClient[0]: >>> sending: metric pmon\_cdr\_calls\_in\_progress [size=1]

INFO: EsClient[0]: <<< POST http://elasticvm:9200/\_bulk - 200

INFO: EsClient[0]: >>> sending: metric pmon\_cdr\_time\_before\_answer [size=20]

INFO: EsClient[0]: <<< POST http://elasticvm:9200/\_bulk - 200

INFO: EsClient[0]: >>> sending: metric pmon\_cdr\_time\_before\_ringing [size=20]

INFO: EsClient[0]: <<< POST http://elasticvm:9200/\_bulk - 200

INFO: EsClient[0]: >>> sending: metric pmon\_cdr\_calls\_by\_duration [size=20]

INFO: EsClient[0]: <<< POST http://elasticvm:9200/\_bulk - 200

INFO: EsClient[0]: >>> sending: metric pmon\_cdr\_cg\_calls [size=11297]

INFO: EsClient[0]: <<< POST http://elasticvm:9200/\_bulk - 200

INFO: EsClient[0]: >>> sending: metric pmon\_cdr\_bg\_calls [size=480]

INFO: EsClient[0]: <<< POST http://elasticvm:9200/\_bulk - 200

INFO: EsClient[0]: >>> sending: metric pmon\_cdr\_supplementary\_service [size=1620]

INFO: EsClient[0]: <<< POST http://elasticvm:9200/\_bulk - 200

INFO: EsClient[0]: >>> sending: metric pmon\_cdr\_subscriber\_group\_calls [size=3120]

INFO: EsClient[0]: <<< POST http://elasticvm:9200/\_bulk - 200

INFO: EsClient[0]: >>> sending: metric pmon\_voip\_tx\_codec [size=340]

INFO: EsClient[0]: <<< POST http://elasticvm:9200/\_bulk - 200

INFO: EsClient[0]: >>> sending: metric pmon\_voip\_tx\_rx\_codec [size=5780]

INFO: EsClient[0]: <<< POST http://elasticvm:9200/\_bulk - 200

INFO: EsClient[0]: >>> sending: metric pmon\_voip\_rx\_codec [size=340]

INFO: EsClient[0]: <<< POST http://elasticvm:9200/\_bulk - 200

INFO: EsClient[0]: >>> sending: metric pmon\_cdr\_trunk\_calls [size=71432]

INFO: EsClient[0]: <<< POST http://elasticvm:9200/\_bulk - 200

INFO: EsClient[0]: >>> sending: metric pmon\_cdr\_trunk\_calls\_duration [size=21876]

INFO: EsClient[0]: <<< POST http://elasticvm:9200/\_bulk - 200