به نام خدا



دانشگاه صنعتی امیرکبیر

# گزارش پروژه شبکه های کامپیوتری

استاد درس : دکتر صادقیان

دانشجو : کیانا آقاکثیری ۹۸۳۱۰۰۶

تاریخ تحویل : ۱۴۰۱/۰۳/۲۰

نیمسال اول ۱۴۰۱

Prometheus ۴ نوع متریک دارد که هرکدام را بررسی می‌کنیم و مناسب‌ترین را انتخاب می‌کنیم.

### ۱. counter

counter یک متریک تجمعی است که نشان‌دهنده یک شمارنده یکنواخت در حال افزایش است که مقدار آن تنها می‌تواند افزایش یابد یا با restart صفر شود. به عنوان مثال، می توان از یک counter برای نمایش تعداد درخواست های ارائه شده، وظایف تکمیل شده یا خطاها استفاده کرد.
برای نشان دادن مقداری که ممکن است کاهش یابد از شمارنده استفاده نمی‌کنیم. به عنوان مثال، از شمارنده برای تعداد processهای در حال اجرا استفاده نمی‌کنیم و در عوض از Guage استفاده می‌کنیم.

### ۲. Guage

guage متریکی است که یک مقدار عددی واحد را نشان می‌دهد که می‌تواند خودسرانه بالا و پایین برود.
guageها معمولاً برای مقادیر اندازه‌گیری شده مانند دما یا **میزان مصرف فعلی حافظه** استفاده می‌شوند، اما همچنین «counterهایی» که می‌توانند بالا و پایین بروند، مانند تعداد درخواست‌های همزمان، استفاده می‌شوند.

### ۳. histogram

یک هیستوگرام از مشاهدات (مانند مدت زمان درخواست یا اندازه پاسخ) نمونه برداری می‌کند و آنها را در سطل های قابل تنظیم شمارش می‌کند. همچنین مجموع تمام مقادیر مشاهده شده را ارائه می‌دهد.

### ۴. summary

شبیه به histogram، یک خلاصه مشاهدات را نمونه‌برداری می‌کند (مانند مدت زمان درخواست و اندازه پاسخ). در حالی که تعداد کل مشاهدات و مجموع تمام مقادیر مشاهده شده را نیز ارائه می‌دهد. از یک پنجره زمانی نیز استفاده می‌کند.

با توجه به توضیحات فوق، برای داده‌های ما که درصد میزان مصرف فعلی حافظه، درصد میزان مصرف فعلی cpu، و فرکانس cpu می‌باشد، بهترین متریک guage است. و هم‌چنین اینکه این داده‌ها مربوط به کدام client است را از طریق یک id مشخص می‌کنیم که clientها از هم متمایز باشند.

حال خروجی کد را بر روی Prometheus مشاهده می‌کنیم:

برای حالتی که یک client وجود دارد:

```
# HELP python_gc_objects_collected_total Objects collected during gc
# TYPE python_gc_objects_collected_total counter
python_gc_objects_collected_total{generation="0"} 448.0
python_gc_objects_collected_total{generation="1"} 615.0
python_gc_objects_collected_total{generation="2"} 0.0
# HELP python_gc_objects_uncollectable_total Uncollectable object found during GC
# TYPE python_gc_objects_uncollectable_total counter
python_gc_objects_uncollectable_total{generation="0"} 0.0
python_gc_objects_uncollectable_total{generation="1"} 0.0
python_gc_objects_uncollectable_total{generation="2"} 0.0
# HELP python_gc_collections_total Number of times this generation was collected
# TYPE python_gc_collections_total counter
python_gc_collections_total{generation="0"} 90.0
python_gc_collections_total{generation="1"} 8.0
python_gc_collections_total{generation="2"} 0.0
# HELP python_info Python platform information
# TYPE python_info gauge
python_info{implementation="CPython",major="3",minor="6",patchlevel="12",version="3.6.12"} 1.0
# HELP ram_percentage ram percentage per second
# TYPE ram_percentage gauge
ram_percentage{client_id="99999"} 78.9
# HELP cpu_percentage cpu percentage per second
# TYPE cpu_percentage gauge
cpu_percentage{client_id="99999"} 17.2
# HELP cpu_frequency cpu frequency per second
# TYPE cpu_frequency gauge
cpu_frequency{client_id="99999"} 2400.0
```

و می‌بینیم که در زمان‌های مختلف داده‌ها فرق می‌کند:

```
# HELP python_gc_objects_collected_total Objects collected during gc
# TYPE python_gc_objects_collected_total counter
python_gc_objects_collected_total{generation="0"} 448.0
python_gc_objects_collected_total{generation="1"} 615.0
python_gc_objects_collected_total{generation="2"} 0.0
# HELP python_gc_objects_uncollectable_total Uncollectable object found during GC
# TYPE python_gc_objects_uncollectable_total counter
python_gc_objects_uncollectable_total{generation="0"} 0.0
python_gc_objects_uncollectable_total{generation="1"} 0.0
python_gc_objects_uncollectable_total{generation="2"} 0.0
# HELP python_gc_collections_total Number of times this generation was collected
# TYPE python_gc_collections_total counter
python_gc_collections_total{generation="0"} 91.0
python_gc_collections_total{generation="1"} 8.0
python_gc_collections_total{generation="2"} 0.0
# HELP python_info Python platform information
# TYPE python_info gauge
python_info{implementation="CPython",major="3",minor="6",patchlevel="12",version="3.6.12"} 1.0
# HELP ram_percentage ram percentage per second
# TYPE ram_percentage gauge
ram_percentage{client_id="99999"} 77.4
# HELP cpu_percentage cpu percentage per second
# TYPE cpu_percentage gauge
cpu_percentage{client_id="99999"} 19.5
# HELP cpu_frequency cpu frequency per second
# TYPE cpu_frequency gauge
cpu_frequency{client_id="99999"} 2400.0
```

حال یک client دیگر اضافه می‌کنیم و مشاهده می‌کنیم که با id متفاوت در حال دادن خروجی می‌باشد:

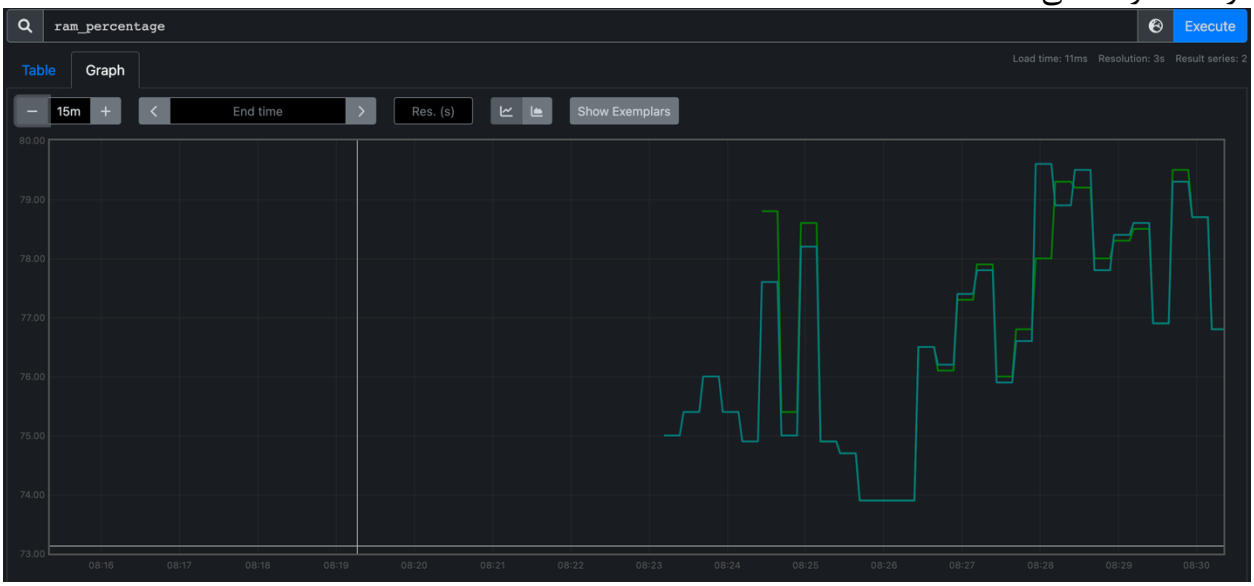
```
# HELP python_gc_objects_collected_total Objects collected during gc
# TYPE python_gc_objects_collected_total counter
python_gc_objects_collected_total{generation="0"} 448.0
python_gc_objects_collected_total{generation="1"} 615.0
python_gc_objects_collected_total{generation="2"} 0.0
# HELP python_gc_objects_uncollectable_total Uncollectable object found during GC
# TYPE python_gc_objects_uncollectable_total counter
python_gc_objects_uncollectable_total{generation="0"} 0.0
python_gc_objects_uncollectable_total{generation="1"} 0.0
python_gc_objects_uncollectable_total{generation="2"} 0.0
# HELP python_gc_collections_total Number of times this generation was collected
# TYPE python_gc_collections_total counter
python_gc_collections_total{generation="0"} 91.0
python_gc_collections_total{generation="1"} 8.0
python_gc_collections_total{generation="2"} 0.0
# HELP python_info Python platform information
# TYPE python_info gauge
python_info{implementation="CPython",major="3",minor="6",patchlevel="12",version="3.6.12"} 1.0
# HELP ram_percentage ram percentage per second
# TYPE ram_percentage gauge
ram_percentage{client_id="99999"} 76.0
ram_percentage{client_id="99998"} 75.6
# HELP cpu_percentage cpu percentage per second
# TYPE cpu_percentage gauge
cpu_percentage{client_id="99999"} 17.3
cpu_percentage{client_id="99998"} 13.8
# HELP cpu_frequency cpu frequency per second
# TYPE cpu_frequency gauge
cpu_frequency{client_id="99999"} 2400.0
cpu_frequency{client_id="99998"} 2400.0
```

که در زمان‌های مختلف هم داده‌ها فرق می‌کنند:

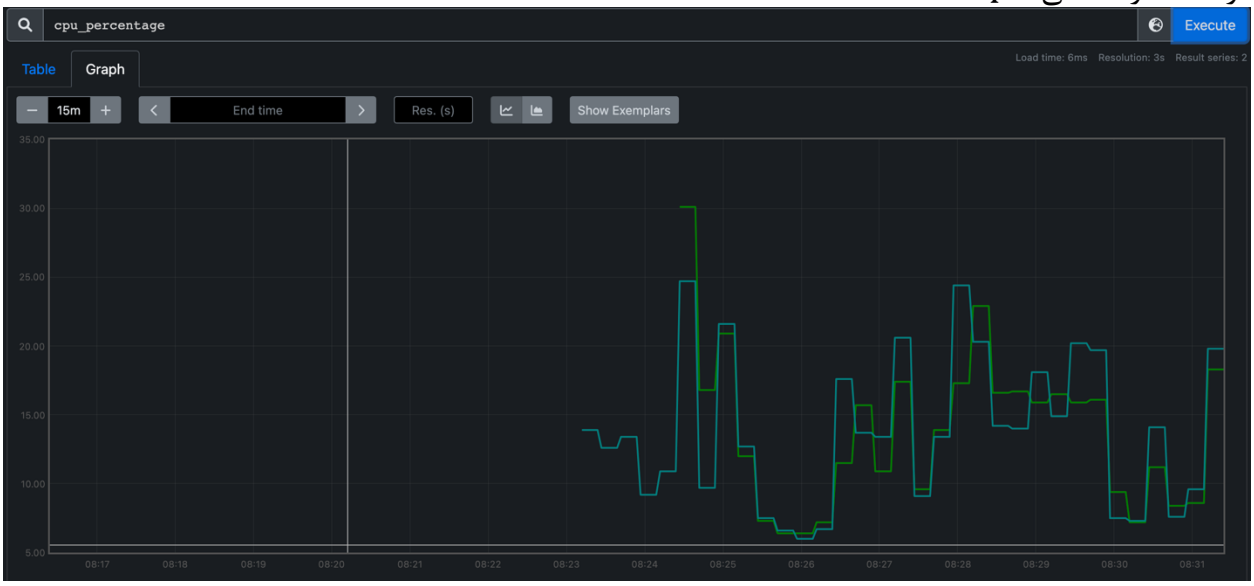
```
# HELP python_gc_objects_collected_total Objects collected during gc
# TYPE python_gc_objects_collected_total counter
python_gc_objects_collected_total{generation="0"} 448.0
python_gc_objects_collected_total{generation="1"} 615.0
python_gc_objects_collected_total{generation="2"} 0.0
# HELP python_gc_objects_uncollectable_total Uncollectable object found during GC
# TYPE python_gc_objects_uncollectable_total counter
python_gc_objects_uncollectable_total{generation="0"} 0.0
python_gc_objects_uncollectable_total{generation="1"} 0.0
python_gc_objects_uncollectable_total{generation="2"} 0.0
# HELP python_gc_collections_total Number of times this generation was collected
# TYPE python_gc_collections_total counter
python_gc_collections_total{generation="0"} 91.0
python_gc_collections_total{generation="1"} 8.0
python_gc_collections_total{generation="2"} 0.0
# HELP python_info Python platform information
# TYPE python_info gauge
python_info{implementation="CPython",major="3",minor="6",patchlevel="12",version="3.6.12"} 1.0
# HELP ram_percentage ram percentage per second
# TYPE ram_percentage gauge
ram_percentage{client_id="99999"} 78.8
ram_percentage{client_id="99998"} 78.6
# HELP cpu_percentage cpu percentage per second
# TYPE cpu_percentage gauge
cpu_percentage{client_id="99999"} 18.3
cpu_percentage{client_id="99998"} 20.9
# HELP cpu_frequency cpu frequency per second
# TYPE cpu_frequency gauge
cpu_frequency{client_id="99999"} 2400.0
cpu_frequency{client_id="99998"} 2400.0
```

در نهایت می‌توانیم یک نمودار از هر کدام از داده‌ها را مشاهده کنیم که در زمان‌های مختلف مقادیر متفاوت داشته‌اند:

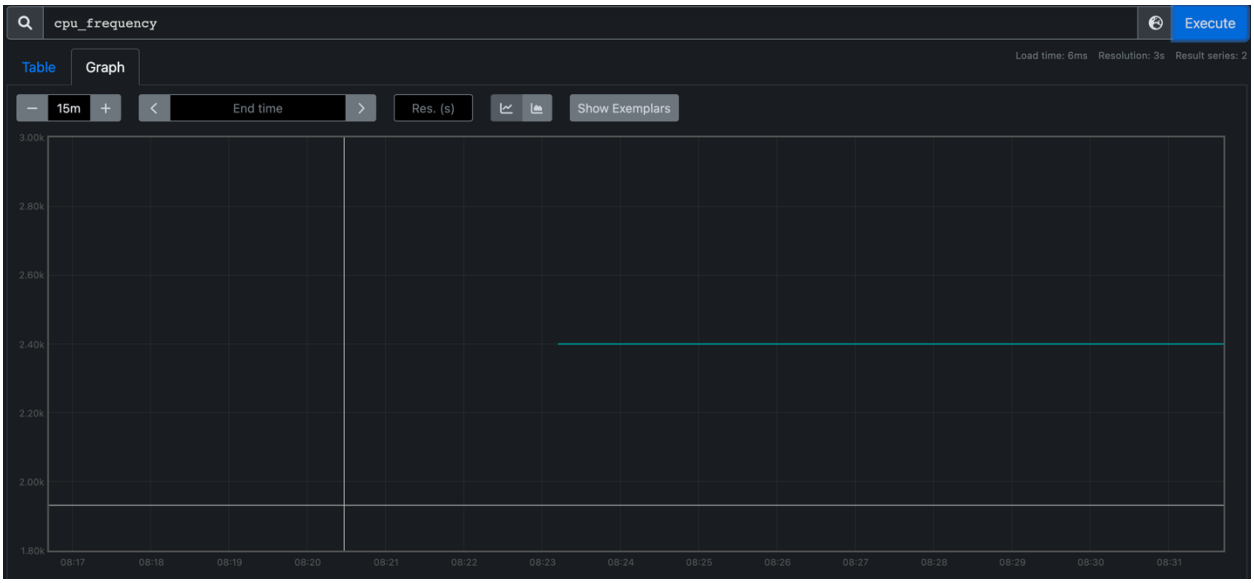
درصد مصرف فعلی حافظه:




درصد مصرف فعلی cpu:



فرکانس cpu:
(که ثابت است)




حال اگر بخواهیم توضیحی درباره کد داشته باشیم:

دوتا فایل پایتون داریم، یکی برای client و دیگری برای server:

Client یک socket می‌سازد، داده‌های مورد نیاز که در کد من درصد میزان مصرف فعلی حافظه و درصد میزان مصرف فعلی cpu و فرکانس cpu است، را می‌گیرد و به server می‌فرستد. هم‌چنین اگر امکان برقراری ارتباط با server مهیا نشد، ...retrying چاپ می‌شود و بعد از ۳ ثانیه دوباره تلاش می‌کند.

Server نیز یک socket می‌سازد، ارتباط با client را برقرار می‌کند و سپس داده های فرستاده شده از client را می‌خواند و برای نشان دادن در Prometheus آماده می‌کند. هم‌چنین اگر داده ای که از client فرستاده می‌شود، خالی باشدو no more data چاپ می‌شود و دوباره داده دریافت می‌کند.

```
'''
create socket and returns the socket
'''


def client(port, IP):
    sock = socket.socket()
    sock.connect((IP, port))  # input = maximum clients
    return sock


def send_data():
    dictionary_data = {}
    dictionary_data['cpu_percentage'] = psutil.cpu_percent(interval=1)
    dictionary_data['ram_percentage'] = psutil.virtual_memory().percent
    dictionary_data['cpu_frequency'] = psutil.cpu_freq()[0]
    return dictionary_data


def send_metrics(client_s):
    while True:
        data = send_data()
        data = json.dumps(data)
        print(data)
        client_s.send(data.encode())


if __name__ == '__main__':
    while True:
        #  if it didnt connect to the server, try to connect again
        try:
            clients = client(8080, '127.0.0.1')
            send_metrics(clients)
        except:
            print("retrying...")
            time.sleep(3)
```

```python
import socket
import json
from _thread import *
from prometheus_client import Gauge, start_http_server

'''
create socket and returns the socket
'''

def server(port, IP):
    sock = socket.socket()
    sock.bind((IP, port))
    sock.listen()  # input = maximum clients
    return sock


'''
accept one client and make connection
'''


def accept(sock):
    connection, client_IP = sock.accept()
    # data_json = connection.recv(1024).decode()  # maximum byte
    # data = json.loads(data_json)
    return client_IP, connection


'''
get data from client and send to prometheus
'''


def get_data(conn, id):
    while True:
        data_json = conn.recv(1024).decode()  # maximum byte
        print(data_json)
        if not data_json:
            print("no more data")
            break
        data = json.loads(data_json)
        ram_percentage.labels(client_id=id).set(data['ram_percentage'])
        cpu_percentage.labels(client_id=id).set(data['cpu_percentage'])
        cpu_frequency.labels(client_id=id).set(data['cpu_frequency'])


ram_percentage = Gauge('ram_percentage', 'ram percentage per second', ['client_id'])
cpu_percentage = Gauge('cpu_percentage', 'cpu percentage per second', ['client_id'])
cpu_frequency = Gauge('cpu_frequency', 'cpu frequency per second', ['client_id'])

if __name__ == '__main__':
    start_http_server(1888)  # runs http server on prometheus
    serv = server(8080, '0.0.0.0')
    id = 100000
    while True:
        client_IP, connection = accept(serv)  # TCP
        print("connected to server")
        id -= 1
        start_new_thread(get_data, (connection, id))
```