



پروژه درس شبکه‌های کامپیوتری

استاد درس: دکتر صادقیان

تهیه و نگارش: امیر بهنام ۹۸۳۱۱۳۳

این پروژه یک برنامه کاربردی مدل **کلاينت-سرور** است که برای **پایش زنده و بلادرنگ مصرف پردازنده (CPU)** یک رایانه طراحی شده است. این سامانه داده‌های عملکردی CPU را جمع‌آوری کرده و آن را به قالب متریک‌های **Prometheus** (یک سامانه قدرتمند مانیتورینگ و پایگاه داده سری‌زمانی) تبدیل می‌کند. این کار به شما امکان می‌دهد مصرف CPU را به مرور زمان نمودار کنید، برای آن اعلان هشدار تنظیم کنید و داشبوردهای مدیریتی (مثلاً با استفاده از Grafana) ایجاد کنید.

نحوه عملکرد: یک breakdown مرحله به مرحله

این سامانه از سه بخش اصلی تشکیل شده است:

۱. **کلاينت (client.py)** - جمع‌آوری‌کننده داده

۲. **سرور (server.py)** - صادرکننده متریک‌ها (اکسپورتر)

۳. **پیکربندی پرومیتئوس (prometheus.yml)** - backend مویتورینگ

در نمودار زیر جریان داده بین این کامپوننت‌ها نشان داده شده است:



۱. کلاینت (جمع آوری کننده داده):

- **هدف:** روی ماشینی که قصد مانیتورینگ آن را دارید اجرا می شود. وظیفه آن اندازه گیری مداوم مصرف CPU و ارسال این داده به سرور است.

• نحوه اجرا:

۱. یک اتصال سوکت **TCP** پایدار به آدرس IP سرور (127.0.0.1) و پورت 8080 برقرار می کند.

۲. در یک حلقه بی نهایت، از کتابخانه **psutil** برای گرفتن میانگین درصد **utilization** CPU در بازه ۱ ثانیه استفاده می کند.

۳. این درصد را در یک شیء **JSON** بسته بندی می کند (مثال "The Percentage Of CPU USAGE" : 12.5}).

۴. این داده **JSON** را از طریق سوکت برای سرور می فرستد و منتظر پاسخ می ماند (پاسخی که دریافت می کند را چاپ می کند).

۲. سرور (پل و اکسپورت تر متریک):

- **هدف:** به عنوان یک پل بین کلاینت و پرومیتئوس عمل می کند. داده خام CPU را دریافت کرده و آن را به قالبی که پرومیتئوس بتواند درک کند تبدیل می نماید.

• نحوه اجرا:

۱. یک سرور سوکت **TCP** روی پورت 8080 راه می اندازد تا به اتصالات ورودی از سمت کلاینت ها گوش دهد.

۲. به طور همزمان، یک سرور **HTTP** روی پورت 8000 با استفاده از **prometheus_client** راه اندازی می کند. این سرور یک **endpoint** به نام **/metrics** ارائه می دهد که پرومیتئوس برای **scrape** کردن (جمع آوری) متریک ها از آن استفاده می کند.

۳. به ازای هر کلاینتی که متصل می شود، سرور یک **thread** جدید برای مدیریت ارتباط بدون مسدود کردن دیگران راه اندازی می کند.

۴. داده JSON را از کلاینت دریافت می‌کند، آن را decode کرده و مقدار درصد CPU را استخراج می‌کند.

۵. یک متریک **Prometheus Gauge** به نام **my_cpu_percents** را با این مقدار جدید به‌روزرسانی می‌کند Gauge. یک متریک است که یک مقدار عددی را نشان می‌دهد که می‌تواند بالا و پایین برود و برای نمایش مصرف CPU کاملاً مناسب است.

۶. این متریک با اطلاعات سوکت کلاینت (آدرس IP و پورت آن) برچسب‌گذاری (label) می‌شود. این قابلیت در آینده به شما امکان می‌دهد چندین کلاینت را به راحتی مانیتور کنید.

• خروجی برنامه در کلاینت و سرور:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python Debug Console + - [ ] [ ] ^ X

Behnam\Downloads\Server2.py
Connect Through ('127.0.0.1', 64858)
PS C:\Users\dr. Behnam\Downloads> c:; cd 'c:\Users\dr. Behnam\Downloads'; & 'C:\Users\dr. Behnam\AppData\Local\Programs\Python\Python38-32\python.exe' 'c:\Users\dr. Behnam\.vscode\extensions\ms-python.python-2021.12.1559732655\pythonFiles\lib\python\debugpy\launcher' '54897' '--' 'c:\Users\dr. Behnam\Downloads\Server2.py'
Connect Through ('127.0.0.1', 54902)
[]
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python Debug Console + - [ ] [ ] ^ X

Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\dr. Behnam> & 'C:\Users\dr. Behnam\AppData\Local\Programs\Python\Python38-32\python.exe' 'c:\Users\dr. Behnam\.vscode\extensions\ms-python-2021.12.1559732655\pythonFiles\lib\python\debugpy\launcher' '61164' '--' 'c:\Users\dr. Behnam\Client2.py'
YOU HAVE BEEN CONNECTED
{"THE PERCENTAGE OF CPU USAGE": 14.6}
{"THE PERCENTAGE OF CPU USAGE": 7.3}
{"THE PERCENTAGE OF CPU USAGE": 5.4}
{"THE PERCENTAGE OF CPU USAGE": 8.5}
{"THE PERCENTAGE OF CPU USAGE": 5.1}
{"THE PERCENTAGE OF CPU USAGE": 33.8}
{"THE PERCENTAGE OF CPU USAGE": 35.7}
{"THE PERCENTAGE OF CPU USAGE": 6.2}
{"THE PERCENTAGE OF CPU USAGE": 33.5}
{"THE PERCENTAGE OF CPU USAGE": 35.7}
{"THE PERCENTAGE OF CPU USAGE": 17.8}
[]
```

```
127.0.0.1:8000 x +
127.0.0.1:8000

# HELP python_gc_objects_collected_total Objects collected during gc
# TYPE python_gc_objects_collected_total counter
python_gc_objects_collected_total{generation="0"} 500.0
python_gc_objects_collected_total{generation="1"} 390.0
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"} 83.0
python_gc_collections_total{generation="1"} 7.0
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="8",patchlevel="10",version="3.8.10"} 1.0
# HELP my_cpu_percents Description of the amount of percentage of cpu usage in gauge
# TYPE my_cpu_percents gauge
my_cpu_percents{endpoint="('127.0.0.1', 54902)",method=" "} 37.4
```

۳. پرومتئوس (مانیتورینگ backend):

- هدف scrape کردن، ذخیره و query کردن متریک‌های ارائه شده توسط سرور.
- پیکربندی (prometheus.yml):
 - فایل پیکربندی یک **scrape job** به نام "my_cpu" تعریف می‌کند.
 - این job به پرومتئوس دستور می‌دهد که متریک‌ها را از هدف localhost:8000 (همان سرور HTTP که توسط server.py راه‌اندازی شده) scrape کند.
 - پرومتئوس به طور خودکار هر ۱۵ ثانیه (مطابق با scrape_interval) endpoint مربوط به metrics/ را بررسی کرده، مقدار فعلی gauge یعنی my_cpu_percents را جمع‌آوری کرده و در پایگاه داده سری‌زمانی خود ذخیره می‌کند.
- خروجی برنامه در Prometheus:

Targets

All Unhealthy Collapse All

my_cpu (1/1 up) [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:8000/metrics	UP	instance="localhost:8000" job="my_cpu"	1.704s ago	342.201ms	

prometheus (1/1 up) [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	2.993s ago	15.600ms	

The screenshot shows the Prometheus web interface in a browser. The address bar shows the URL: 127.0.0.1:9090/graph?g0.expr=my_cpu_percents&g0.tab=1&g0.stacked=0&g0.show_exemplars=0&g0.range_input=1h. The interface includes a navigation bar with 'Prometheus', 'Alerts', 'Graph', 'Status', and 'Help'. Below the navigation bar, there are checkboxes for 'Use local time', 'Enable query history', 'Enable autocomplete', 'Enable highlighting', and 'Enable linter'. The main query input field contains 'my_cpu_percents' and an 'Execute' button. Below the query input, there are tabs for 'Table' and 'Graph'. The 'Table' tab is selected, showing a table with one row of data: my_cpu_percents{endpoint="(127.0.0.1, 62158)", instance="localhost:8000", job="my_cpu", method=""}. The value in the table is 22.6. There is also an 'Add Panel' button at the bottom left.

دلیل استفاده از Gauge :

همانطور که در گزارش شما به درستی اشاره شده است، انتخاب نوع متریک **Gauge** کاملاً عمدی و صحیح بوده است:

- **Gauge در مقابل Counter:** یک Counter فقط برای افزایش طراحی شده است (مثلاً برای شمارش کل درخواست‌ها، خطاها یا کارهای تکمیل شده). از آن همراه با تابع `rate()` برای فهم نرخ وقوع رویداد استفاده می‌شود. یک Gauge می‌تواند خودسرانه کم و زیاد شود، که این دقیقاً ماهیت درصد CPU utilization است (مثلاً `14.6%`, `7.3%`, `35.7%`).

- **Counter در مقابل Gauge:** یک Counter فقط برای افزایش طراحی شده است (مثلاً برای شمارش کل درخواست‌ها، خطاها یا کارهای تکمیل شده). از آن همراه با تابع `rate()` برای فهم نرخ وقوع رویداد استفاده می‌شود. یک Gauge می‌تواند خودسرانه کم و زیاد شود، که این دقیقاً ماهیت درصد CPU utilization است (مثلاً `14.6%`, `7.3%`, `35.7%`).

خروجی و اعتبارسنجی:

همانطور که در خروجی ترمینال و UI پرومیتئوس نشان داده شده است، پروژه با موفقیت اجرا شد:

- **خروجی ترمینال:** کلاینت با موفقیت متصل شد و مقادیر دائماً در حال تغییر مصرف CPU را چاپ کرد (مانند `14.6`, `7.3`, `5.4`, `35.7`).

- **UI پرومیتئوس:** متریک `my_cpu_percents` با موفقیت scrape شد و در رابط وب پرومیتئوس قابل پرس و جو است. برجسب‌ها نشان می‌دهند که این متریک به درستی با job ای به نام `my_cpu` و آدرس کلاینت مرتبط شده است.

جمع‌بندی نهایی:

در اصل، این پروژه یک pipeline ایجاد می‌کند:

→ HTTP Endpoint → Prometheus Gauge → کلاینت → سوکت → سرور → CPU مصرف سرور پرومیتئوس

وقتی داده‌ها در پرومیتئوس قرار گرفتند، برای ساخت داشبورد، تحلیل روندها و تنظیم هشدارهایی که در صورت تجاوز میزان استفاده از CPU از یک آستانه مشخص برای یک دوره زمانی خاص، به شما اطلاع دهند، در دسترس قرار می‌گیرند..