پروژه درس شبکه‌های کامپیوتری

استاد درس: دکتر صادقیان

تهیه و نگارش: امیر بهنام ۹۸۳۱۱۳۳

این پروژه یک برنامه کاربردی مدل **کلاینت–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

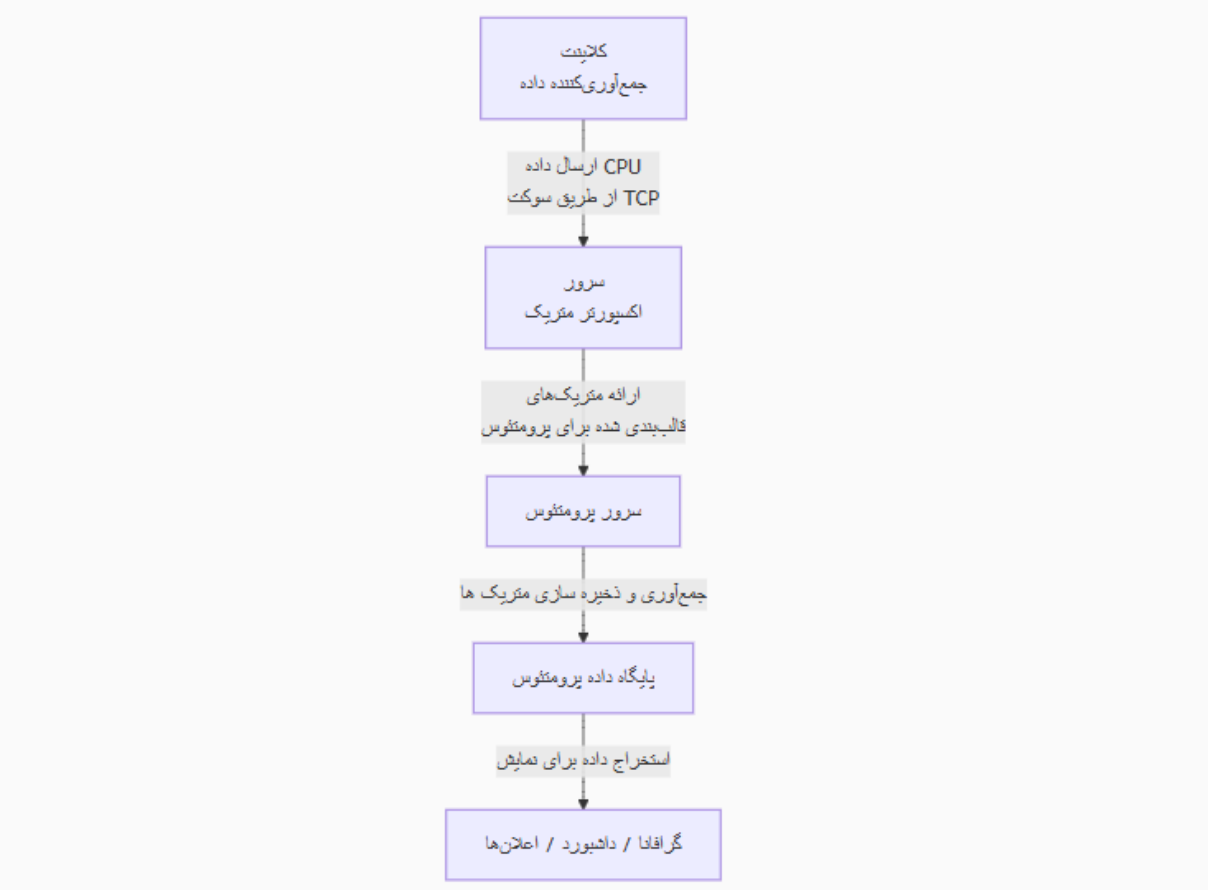
این سامانه از سه بخش اصلی تشکیل شده است:

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

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

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

در نمودار زیر جریان داده بین این کامپوننت ها نشان داده شده است:



۱. کلاینت (جمع‌آوری‌کننده داده):

- **هدف:** روی ماشینی که قصد مانیتورینگ آن را دارید اجرا می‌شود. وظیفه آن اندازه‌گیری مداوم مصرف CPUو ارسال این داده به سرور است.
- **نحوه اجرا:**

۱. یک **اتصال سوکت TCP** پایدار به آدرس IP سرور (127.0.0.1) و پورت8080 برقرار می‌کند.

۲. در یک حلقه بی‌نهایت، از کتابخانه psutil برای گرفتن **میانگین درصد utilization CPU** در بازه ۱ ثانیه استفاده می‌کند.

۳. این درصد را در یک شیء JSON بسته‌بندی می‌کند (مثال {"The Perecentage Of CPU USAGE" : 12.5}).

٤. این داده JSON را از طریق سوکت برای سرور می‌فرستد و منتظر پاسخ می‌ماند (پاسخی که دریافت می‌کند را چاپ می‌کند).

۲. سرور (پل و اکسپورتر متریک):

- **هدف :**به عنوان یک پل بین کلاینت و پرومتئوس عمل می‌کند. داده خام CPU را دریافت کرده و آن را به قالبی که پرومتئوس بتواند درک کند تبدیل می‌نماید.
- **نحوه اجرا:**

۱. یک **سرور سوکت TCP** روی پورت 8080 راه می‌اندازد تا به اتصالات ورودی از سمت کلاینت‌ها گوش دهد.

۲. **به طور همزمان**، یک **سرور HTTP** روی پورت 8000با استفاده از prometheus_client راه‌اندازی می‌کند. این سرور یک endpoint به نام /metricsارائه می‌دهد که پرومتئوس برای scrapeکردن (جمع‌آوری) متریک‌ها از آن استفاده می‌کند.

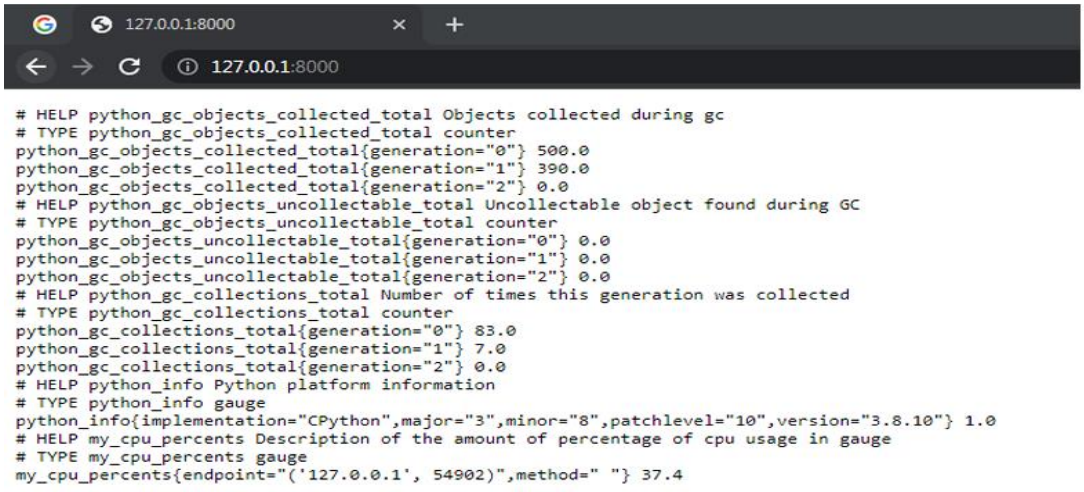
۳. به ازای هر کلاینتی که متصل می‌شود، سرور یک thread جدید برای مدیریت ارتباط بدون مسدود کردن دیگران راه‌اندازی می‌کند.

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

٥. 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.

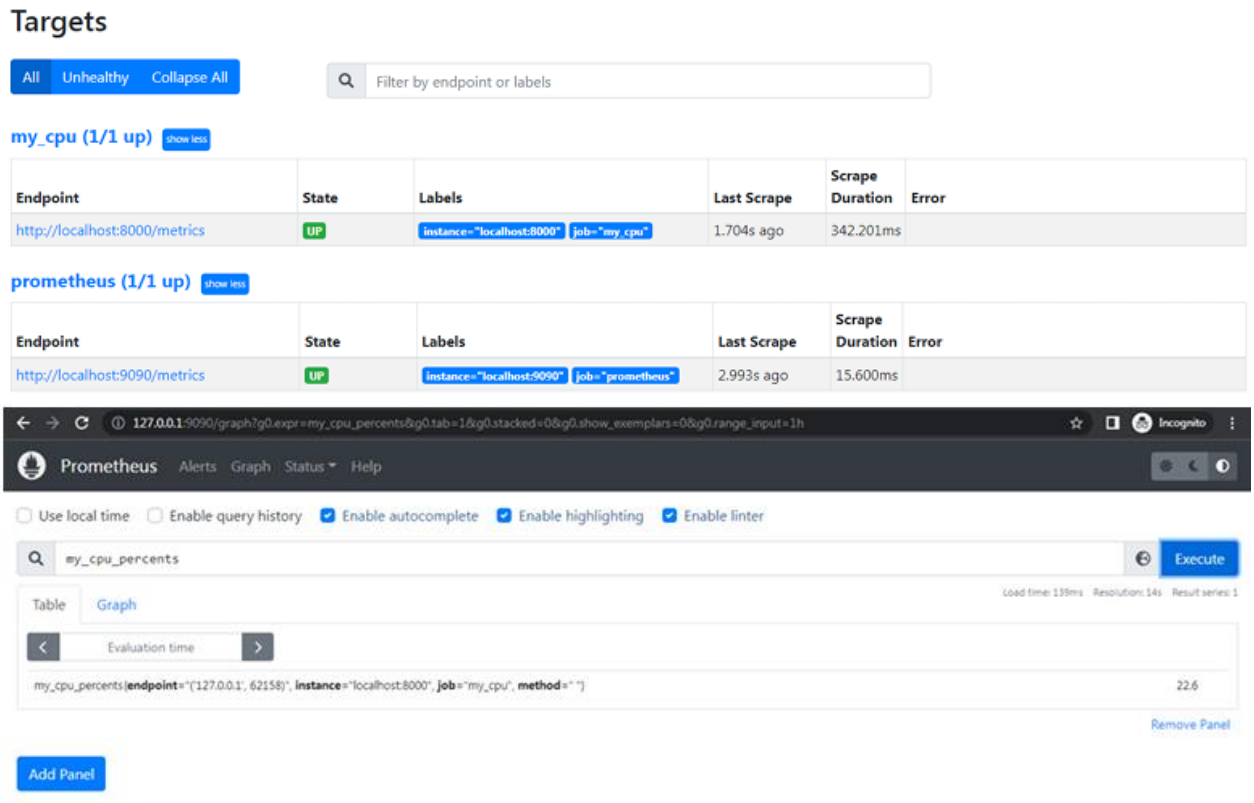
٦. این متریک با اطلاعات سوکت کلاینت (آدرس IP و پورت آن) برچسب‌گذاری (label) می‌شود. این قابلیت در آینده به شما امکان می‌دهد چندین کلاینت را به راحتی مانیتور کنید.

- **خروجی برنامه در کلایت و سرور:**



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**:



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 Counterفقط 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 utilization CPU است (مثلاً ۱۴.۶٪, ۷.۳٪, ۳۵.۷٪).
- **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 utilization CPU است (مثلاً ۱۴.۶٪, ۷.۳٪, ۳۵.۷٪).

خروجی و اعتبارسنجی:

همانطور که در خروجی ترمینال و UI پرومتئوس 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 ۱۴.۶, ۷.۳, ۵.۴, ۳۵.۷).
- **UI پرومتئوس**: متریک my_cpu_percents با موفقیت scrape شد و در رابط وب پرومتئوس قابل پرس و جو است. برچسب‌ها 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:

CPUمصرف → سرور → سوکت → کلاینت → Prometheus Gauge → HTTP Endpoint → سرور پرومتئوس

وقتی داده‌ها در پرومتئوس 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..