گزارش فعالیت پنجم

علیرضا قدوسی نژاد

توضیح کلی کد

در این برنامه، با هدف مقایسه‌ی عملکرد اجرای هم‌زمان در سیستم‌های توزیع‌شده، سه روش پیاده‌سازی multithreadingبررسی شده‌اند:

روش سفارشی باthreading.Thread

یک پیاده‌سازی دستی برای اجرای هم‌زمان چند کارگر (worker) به کمک کلاس.Thread

استفاده از ThreadPoolExecutor از کتابخانهconcurrent.futures

اجرای بهینه‌تر و ساده‌تر تسک‌ها در قالب thread با استفاده از API آماده پایتون.

استفاده ازProcessPoolExecutor

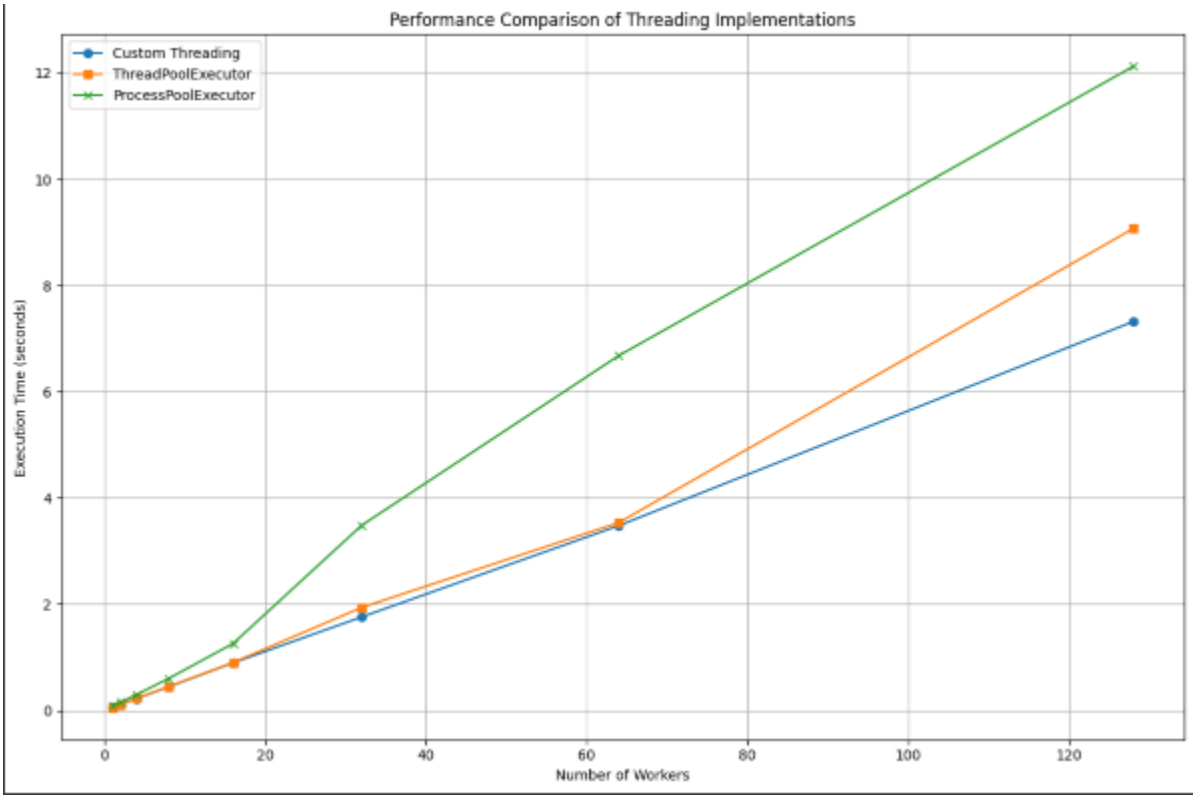
پیاده‌سازی multiprocessing برای کارهای CPU-Bound و بررسی تاثیر استفاده از چند پردازش.

شبیه‌سازی سرویس‌هایApache-like

شبیه‌سازی مدل سرویس‌محور با ارسال درخواست به API (با استفاده ازhttpbin) برای مقایسه با روش‌های قبلی.

در هر روش، زمان اجرای مجموع کارها برای تعداد مختلفی از کارگرها (مثل 1، 2، 4، ... تا 128) اندازه‌گیری شده و نمودارها و جداول برای مقایسه‌ی کارایی تولید شده‌اند. همچنین، برای تسک‌های شبکه‌ای-I/O) (boundو محاسباتی (CPU-bound) به صورت جداگانه آزمایش صورت گرفته است.

نمودار ۱



عملکرد وظایف CPU-Bound (محدوده وسیع تعداد کارگرها)

این نمودار، عملکرد سه پیاده‌سازی مختلف threading را برای وظایف CPU-Bound با تعداد کارگرهای بین 1 تا 128 مقایسه می‌کند:

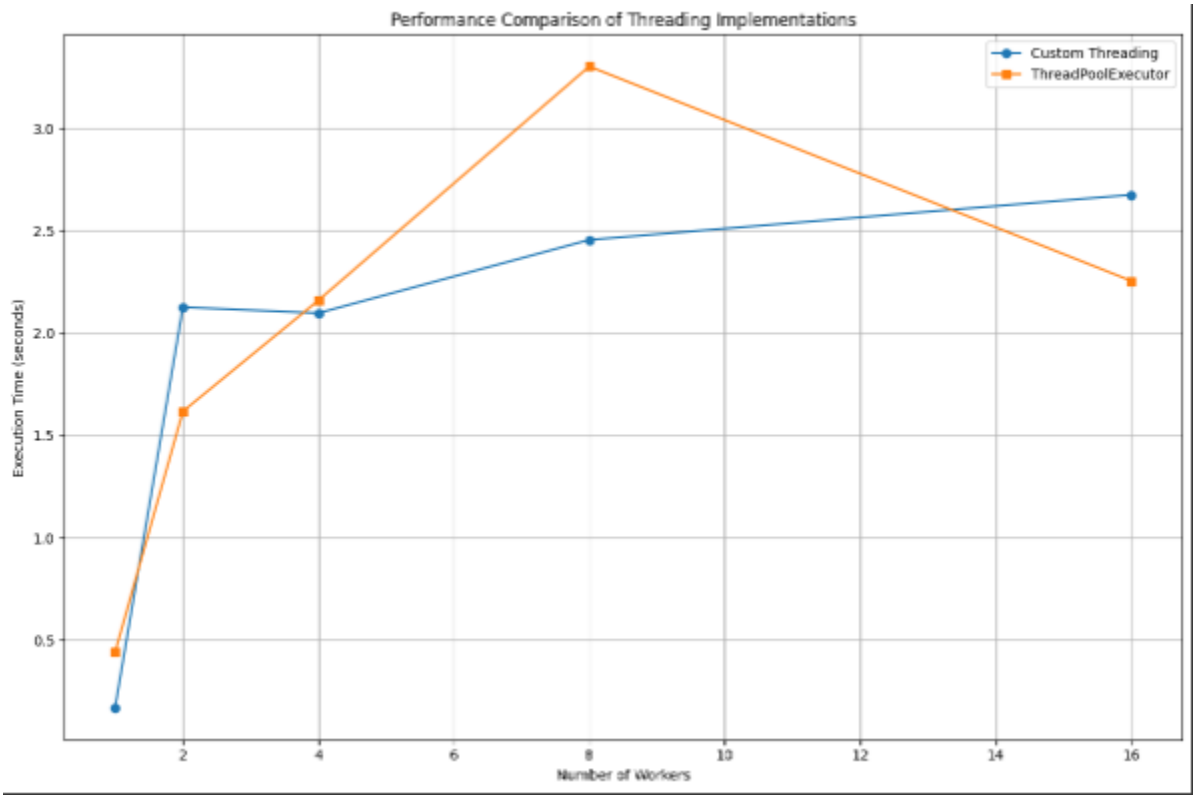
ProcessPoolExecutor (خط سبز) بدترین عملکرد را نشان می‌دهد و با افزایش تعداد کارگرها، زمان اجرا به‌سرعت افزایش یافته و در 128 کارگر به حدود 12 ثانیه می‌رسد.

ThreadPoolExecutor (خط نارنجی) عملکرد بهتری نسبت به روش مبتنی بر پردازش دارد و در 128 کارگر به حدود 9 ثانیه می‌رسد.

Custom Threading (خط آبی) بهترین عملکرد را برای وظایف CPU-Bound دارد و در بیشترین تعداد کارگر (128) زمان اجرای حدود 7.5 ثانیه را حفظ می‌کند.

نتیجه کلیدی: برای وظایف پردازشی سنگین، پیاده‌سازی سفارشی threading عملکرد بهتری از دو روش دیگر دارد، که احتمالاً به‌دلیل سربار کمتر آن است. عملکرد ضعیف ProcessPoolExecutor نیز نشان می‌دهد که سربار مربوط به ایجاد و مدیریت فرآیندها، بیشتر از مزایای بالقوه پردازش موازی برای این وظایف خاص است.

نمودار ۲



عملکرد وظایف CPU-Bound (محدوده کم کارگرها)

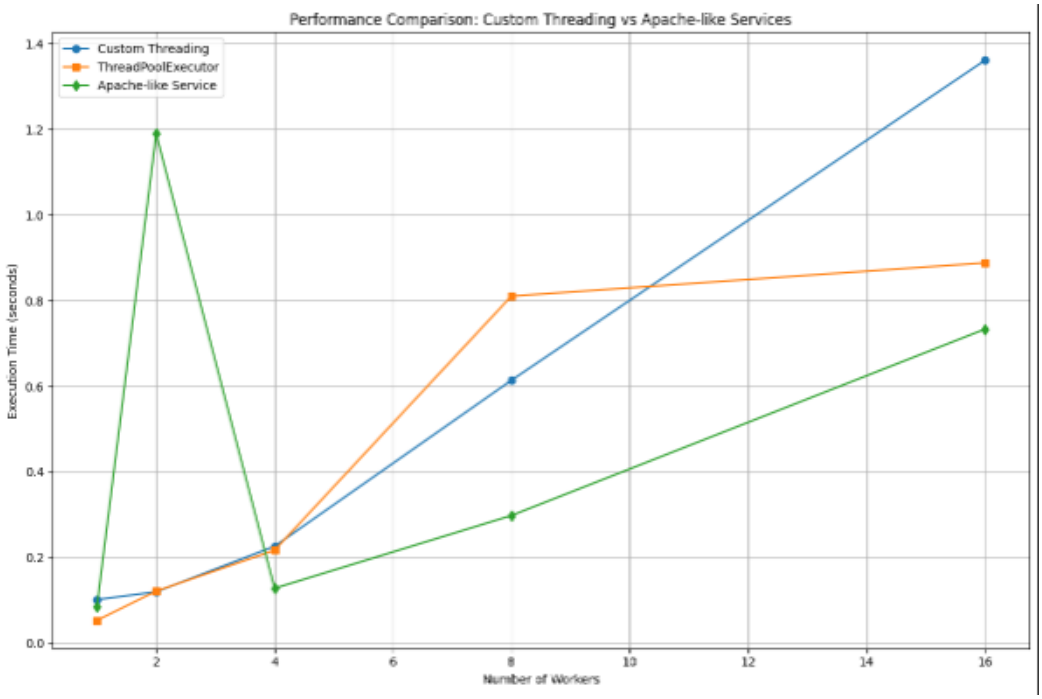
این نمودار، عملکرد پیاده‌سازی‌ها را در محدوده کوچک‌تری از تعداد کارگر (1 تا 16) برای وظایف CPU-Bound بررسی می‌کند:

ThreadPoolExecutor (خط نارنجی) عملکرد ناپایداری دارد و در ۸ کارگر یک افزایش ناگهانی تا حدود ۳.۳ ثانیه مشاهده می‌شود، اما در ۱۶ کارگر به حدود ۲.۲ ثانیه بهبود می‌یابد

Custom Threading (خط آبی) رفتار مقیاس‌پذیری قابل پیش‌بینی‌تری دارد و از ۱ تا ۲ کارگر افزایش تندی دارد، سپس به‌تدریج تا حدود ۲.۷ ثانیه در ۱۶ کارگر رشد می‌کند.

نتیجه کلیدی: در مقیاس کم، رفتار عملکردی پیچیده‌تر است. ThreadPoolExecutor در ۸ کارگر عملکرد بدتری دارد ولی در ۱۶ کارگر بهتر می‌شود. این نشان می‌دهد که ممکن است این اجراگر دارای بهینه‌سازی‌هایی باشد که در مقیاس‌های خاصی فعال می‌شوند.

نمودار ۳



مقایسه با سرویس‌های شبیه به Apache

این نمودار عملکرد threading سفارشی، ThreadPoolExecutor و یک سرویس شبیه به Apache را با ۱ تا ۱۶ کارگر مقایسه می‌کند:

Custom Threading (خط آبی) در تعداد کارگرهای بالا بدترین عملکرد را دارد و زمان اجرا به‌صورت خطی تا حدود ۱.۳۵ ثانیه در ۱۶ کارگر افزایش می‌یابد.

ThreadPoolExecutor (خط نارنجی) مقیاس‌پذیری بهتری دارد و پس از ۸ کارگر زمان اجرا را حدود ۰.۹ ثانیه ثابت نگه می‌دارد.

Apache-like Service (خط سبز) بهترین عملکرد را در تعداد کارگرهای بالا دارد و زمان اجرا را در حدود ۰.۷ ثانیه حفظ می‌کند، به‌جز یک جهش غیرعادی در ۲ کارگر.

نتیجه کلیدی: شبیه‌سازی سرویس Apache عملکرد بهتری در وظایف توزیع‌شده دارد، مخصوصاً با افزایش تعداد کارگر. این نشان‌دهنده مزایای استفاده از سیستم‌های توزیع‌شده تخصصی مانند Apache برای بارهای کاری هم‌زمان است. جهش اولیه ممکن است مربوط به سربار راه‌اندازی اتصال باشد که با افزایش کارگر جبران می‌شود.

نتیجه‌گیری کلی

نوع وظیفه اهمیت دارد: انتخاب مناسب threading به‌شدت به نوع وظیفه (CPU یا توزیع‌شده/شبکه‌ای) وابسته است.

مزایا و معایب Custom Threading: این روش برای وظایف CPU-Bound عالی عمل می‌کند ولی در وظایف توزیع‌شده مقیاس‌پذیری ضعیفی دارد.

پایداری ThreadPoolExecutor: این روش در اکثر سناریوها عملکردی باثبات و قابل اتکا دارد و انتخاب مناسبی برای استفاده عمومی است.

مزیت سرویس‌های Apache: برای وظایف توزیع‌شده، سرویس‌های شبیه به Apache مقیاس‌پذیری بسیار خوبی ارائه می‌دهند.

رفتار مقیاس‌پذیری: همه‌ی پیاده‌سازی‌ها رفتار غیرخطی دارند و عملکرد با افزایش تعداد کارگر به شکل متفاوتی افت می‌کند. بنابراین، انتخاب تعداد کارگر بهینه برای هر پیاده‌سازی بسیار مهم است.