**بررسی روند اجرای درخواست ها در اپلیکیشن میکروسرویس** **DeathStarBench/social network**

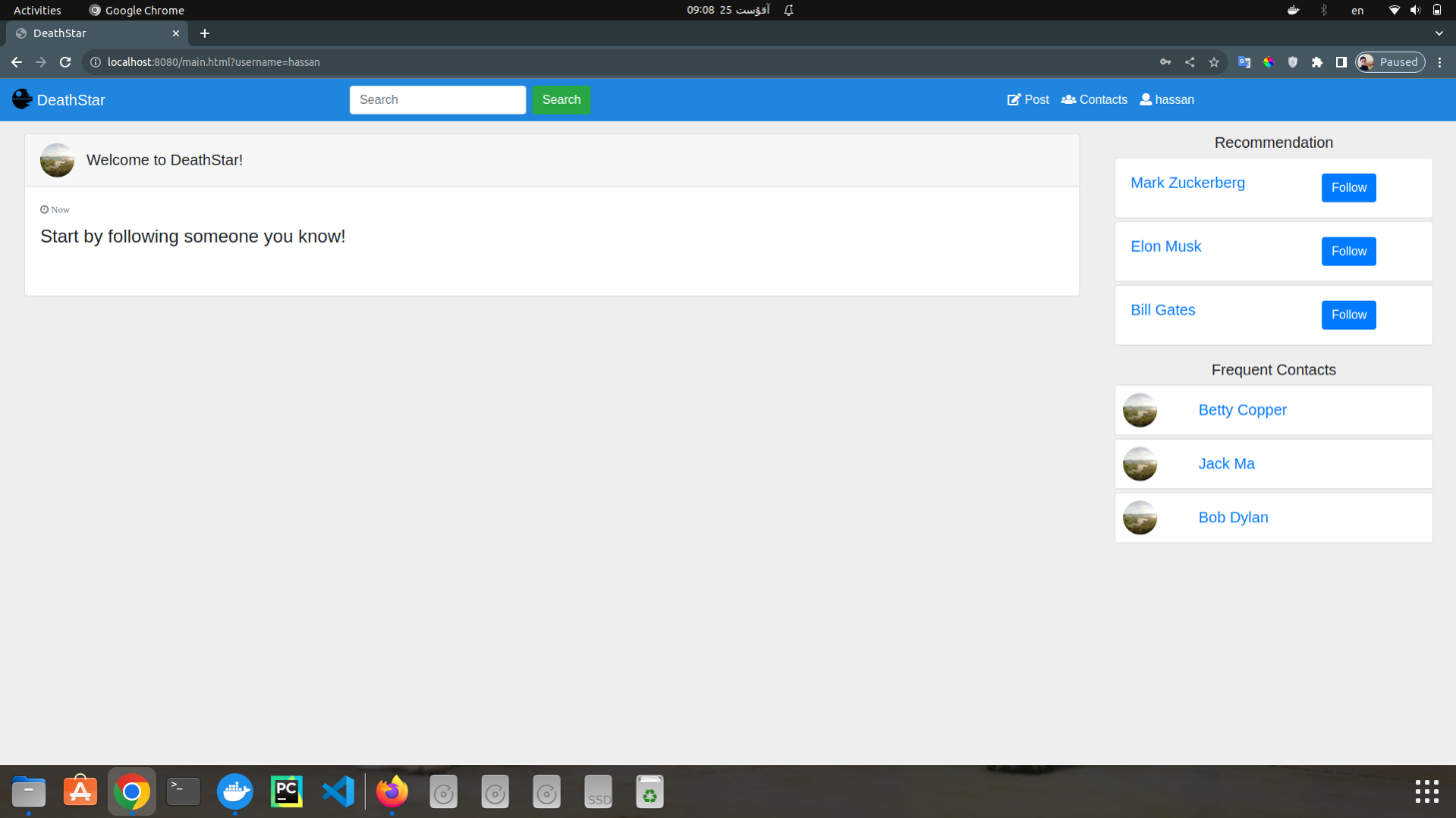
1. دریافت و اجرای پروژه با استفاده از docker-compose

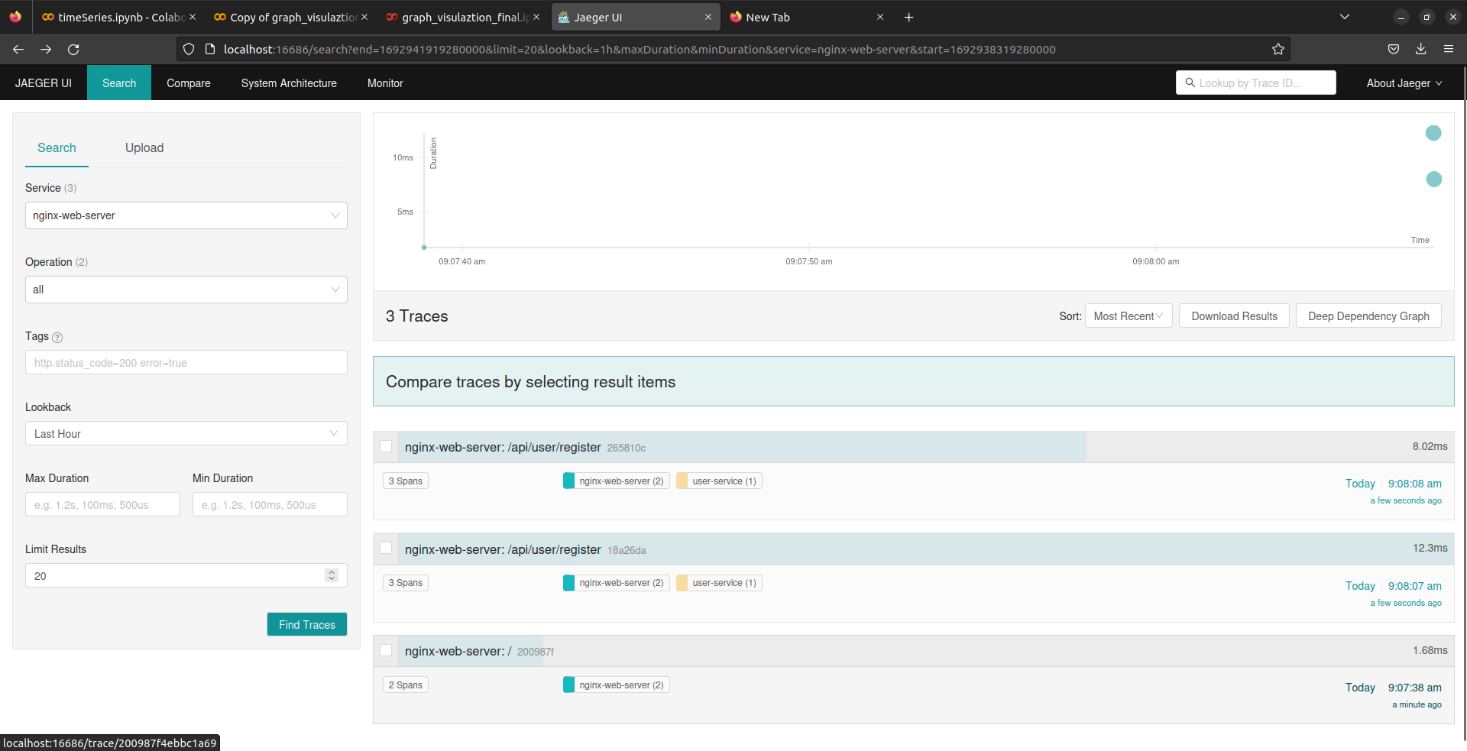
<https://github.com/delimitrou/DeathStarBench/tree/master/socialNetwork>

پروژه از لینک گیت هاب دانلود شده و پس از نصب پیش نیاز ها با کمک docker-compose اجرا شده است.

پیش نیازهای اجرا پروژه

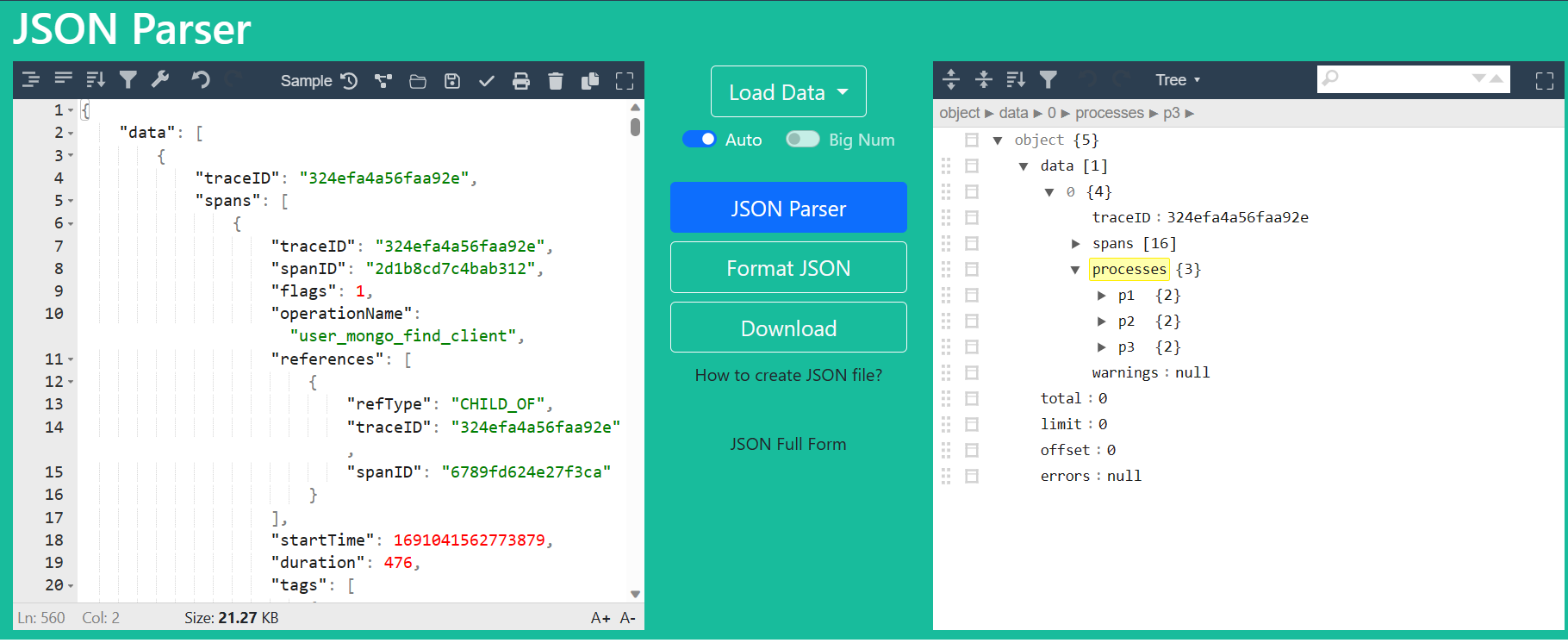
* Docker
* Docker-compose
* Python 3.5+ (with asyncio and aiohttp)
* libssl-dev (apt-get install libssl-dev)
* libz-dev (apt-get install libz-dev)
* luarocks (apt-get install luarocks)

**نمایی از اجرای پروژه** 



**بررسی درخواست ها به صورت جداگانه و رسم نمودار سری زمانی و گراف همسایگی برای هر درخواست**

در این پروژه از jaeger برای نمایش trace های هر درخواست استفاده شده است و در پورت 16686 می توانید به این ابزار دسترسی داشته باشید ، می توانید trace ها رو برای انواع سرویس ها مشاهده کنید ، فایل جیسون هر کدام را دانلود کنید . نمودار زمان اجرای اون درخواست رو مشاهده کنید ،گراف DAG برای هر درخواست قابل مشاهده است و ...

 یک trace که بیانگر اجرای یک درخواست است را دانلود کرده و به بررسی آن پرداخته شده و سعی شده بخش های مهم وکاربردی که روند اجرای درخواست را به خوبی نمایش می دهند به دست بیاوریم و در گراف و نمودار نمایش دهیم به شکلی که با دیدن گراف و نمودار دید خوبی نسبت به درخواست ارسال شده به دست بیاوریم .

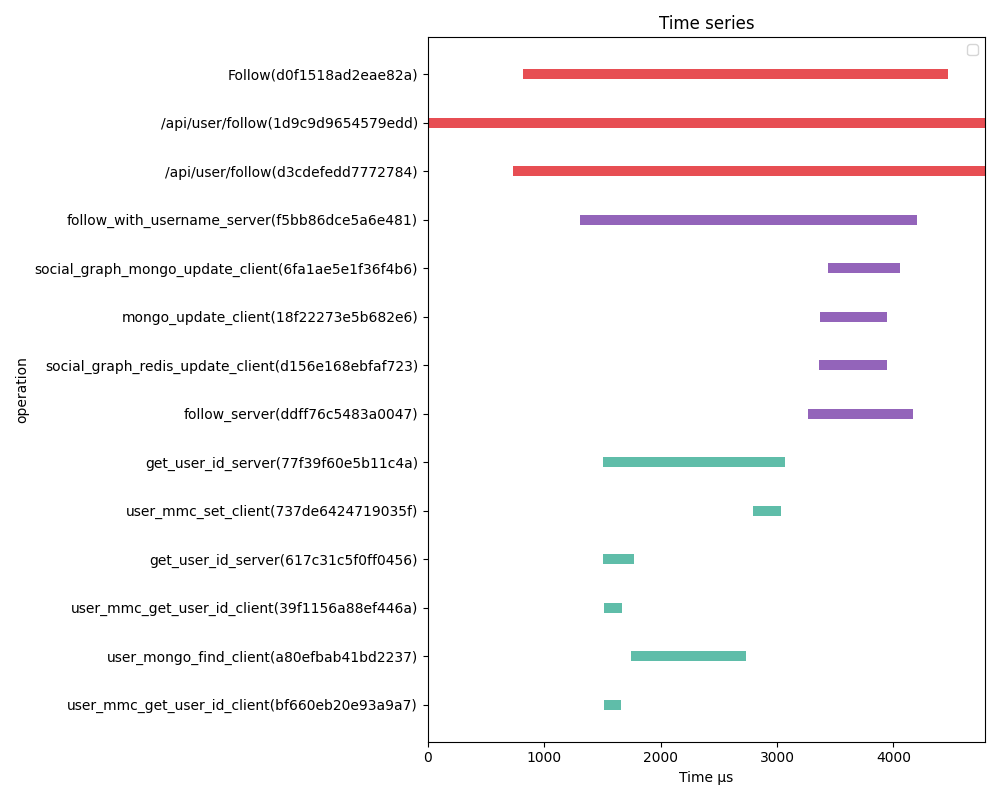
با بررسی جزییات فایل جیسون روند درخواست می توان اطلاعات زیر را برداشت کرد :

**data:**

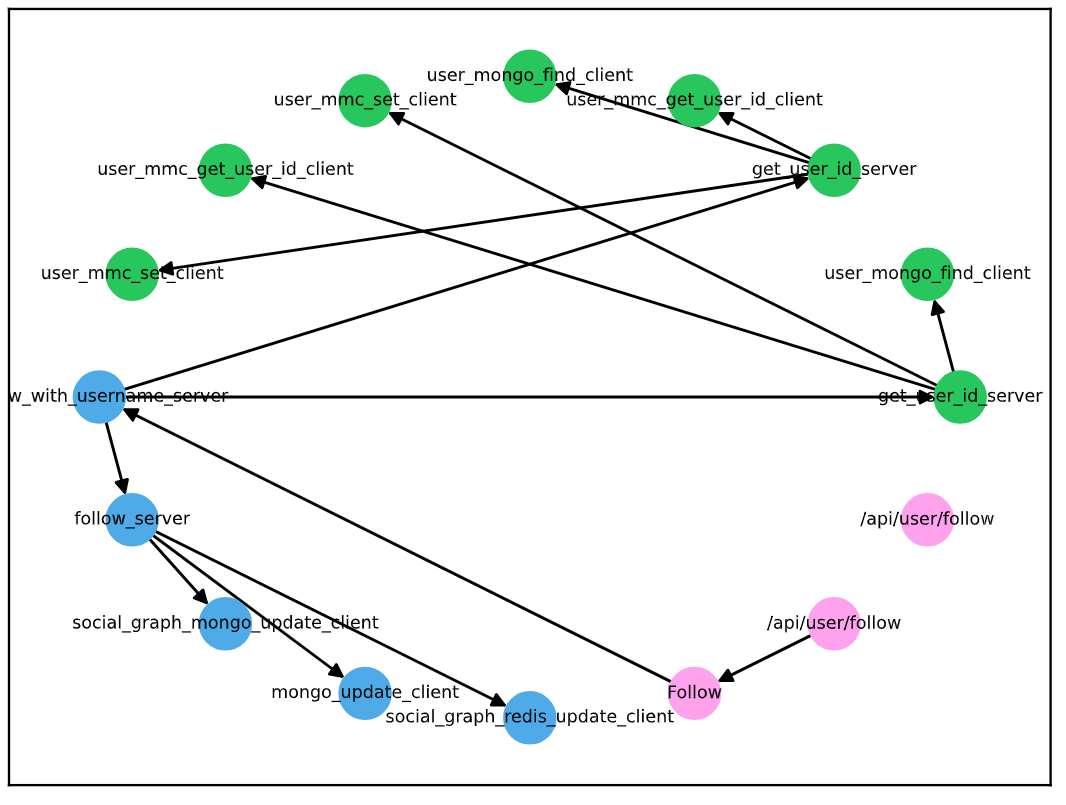
* شامل span هایی است که در روند اجرای درخواست فراخونی شده اند
* شامل traceId که id درخواست ارسال شده رو مشخص می کند .
* Processes که سرویس هایی که برای درخواست فراخوانی شده اند را مشخص میکند.

برای هر span نیز اطلاعاتی در نظر گرفته شده شامل :

* operationName: که شامل نام عملیاتی که انجام می شود است .
* refrences: روابط span را نشان می دهد و از این فیلد متوجه می شویم که هر span می تواند parent یا child دیگری باشد .
* spanId
* duration که شامل مدت زمان اجرای span است
* startTime شامل زمان شروع span می شود

* 1. **برای نمایش درخواست ها از نمودار time\_series استفاده شده که اطلاعات زیر را می توان با دیدن نمودار به دست آورد :**
* مدت زمان اجرای کل درخواست
* زمان شروع و پایان هر span
* ترتیب اجرای span ها برای هر درخواست
* سرویس های مختلف اجرا شده برای هر درخواست ( هر خط افقی بیانگر یک span که زمان شروع و پایان آن در در نمودار مشخص است و spanهایی که متعلق به یک سرویس هستند با رنگ یکسان نمایش داده شده اند.
  1. **نمایش به شکل گراف**

در این گراف سعی شده بر اساس رابطه child/parent که بین span ها وجود دارد گرافی رسم شود به این شکل که از parent به سمت child یال رسم می شود .



برای بررسی درستی کدهای نوشته شده که این نمودار ها حاصل اجرای این کدها هستند نمودارها که از روی فایل json درخواست رسم شده اند با نمودارهای رسم شده در jaeger مقایسه شده و مطابقت داشته اند . هدف در این فاز بررسی درخواست ها ، نمایش بهتر و روشهای visualize کردن درخواست ها بوده است .

**بررسی اجرای درخواست ها به شکل همزمان در یک مدت زمان مشخص**

از آن جایی که در یک اپلیکیشن صرفا یک نوع درخواست ارسال نمیشود و در یک مدت زمان ، درخواست های متنوعی ارسال می شود به دنبال روشی بودیم که بتوانیم چندین درخواست متنوع را دریک بازه ی زمانی ارسال کنیم . با بررسی اپلیکیشن در یکی از اسکریپت های نوشته شده در پروژه این قابلیت وجود دارد و می توانیم از این اسکریپت استفاده کنیم .

با اجرای command زیر می توانیم تعریف کنیم که در مدت زمان مشخص چه تعدادی درخواست ارسال شود.

برای اجرای دستور باید در مسیر socialNetwork وارد شود .

../wrk2/wrk -D exp -t <num-threads> -c <num-conns> **-d <duration**> -L -s ./wrk2/scripts/social-**network/compose-post.lua** http://localhost:8080/wrk2-api/post/compose -R **<reqs-per-sec**>

با بررسی پروژه گیت هاب متوجه شدیم که برای پروژه 4 اسکریپت نوشته شده و 3 اسکریپت مربوط به یک نوع درخواست هستند و یک [**script mixed-workload**](https://github.com/delimitrou/DeathStarBench/blob/master/socialNetwork/wrk2/scripts/social-network/mixed-workload.lua) داشته که هر درخواست را به شکل رندوم از 3 نوع درخواست دیگر ارسال می کند و می توانیم در مدت زمان مشخص درخواست های متنوعی داشته باشیم که همان چیزی است که نیاز داریم . با توجه به اسکریپتی که تعریف شده می توانیم در بازی ی زمانی مد نظر به شکل رندوم از 3 نوع درخواست همزمان استفاده کنیم به شکلی که برای هر درخواست با احتمالی که برای آن درخواست در نظر میگیریم از آن درخواست ارسال می شود برای مثال می توانیم برای درخواست اول 60 درصد ، دومین نوع درخواست 30 درصد و برای درخواست سوم احتمال 10 درصد را تنظیم کنیم .

پارامترهای دستور عبارت است از :

-D بیانگر توزیع ارسال درخواستها که در این exponential گرفته شده ( می توانیم از uniform استفاده کنیم )

-t بیانگر number of threads to use

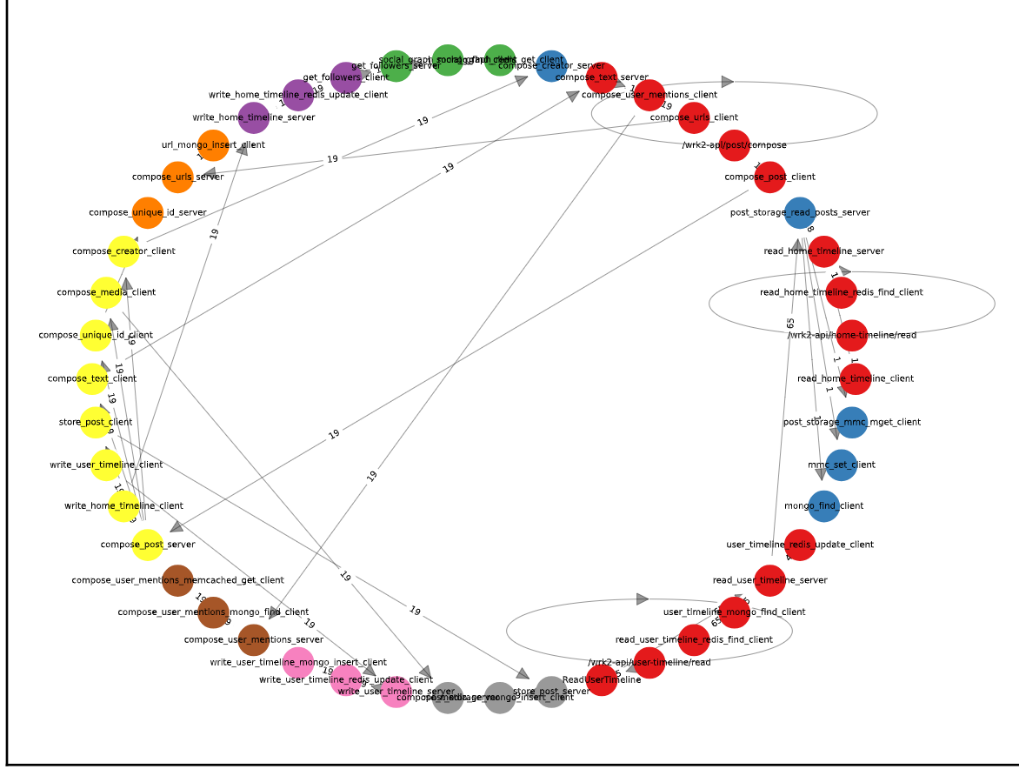
-c بیانگر number of concurrent connections

-d duration of the test می توانیم براساس ثانیه عدد بدهیم یا به صورت ( 10s ، 2m )

-s script that customizes the request load

-R Determines the rate of requests per second to target the server

**نمایش تعداد تکرار ها و فراخوانی های parent/child در درخواست های ارسال شده**

به دنبال این هستیم که بررسی کنیم در مدت زمان ارسال شده و درخواست های ارسال شده چه تعداد فراخوانی تکراری بین span ها وجود دارد . تلاش میکنیم که درخواست ها را براساس رابطه ی parent/child روی گراف نمایش دهیم و به این شکل که هر بار که رابطهparent به child داریم اگر از قبل یال آن روی گراف باشد وزن یال را اضافه میکنیم تا تعداد تکرارهای parent/child را روی اجرای درخواست مشاهده کنیم . ( کد graphForMultipleRequests.ipynb )

1. **بررسی درخواست های ارسال شده در مدت زمان طولانی و گرفتن خروجی به شکل time series در بازه های مشخص**

به دنبال این هستیم که در مدت زمان زیاد و با حجم دیتای که واقعی تر به نظر بیاد یک دیتا ست ایجاد کنیم .

ابتدا باید اپلیکیشن رو اجرا کنیم و فایل trace حاصل از اجرای درخواست ها را به دست بیاوریم ( چون در مدت زمان طولانی نیاز به اجرای اسکریپت داریم باید روی سرور اپلیکیشن رو به حالتی اجرا کنیم که با خارج شدن از سرور دستور اجرای اسکریپت در حال اجرا باقی بماند .)

( برای اجرای اسکریپت و ذخیره کردن آن در فایل result\_requests.json از کدهای زیر استفاده شده است که نتیجه برای اجرای 8 ساعت و تعداد 20 درخواست در ثانیه حجم فایل trace ، 600 مگابایت شده است

nohup ../wrk2/wrk -D exp -t 1 -c 1 -d 28800 -L -s ./wrk2/scripts/social-network/mixed-workload.lua http://localhost:8080 -R 20

curl "http://localhost:16686/api/traces?limit=200000000&lookback=24h&maxD uration&minDuration&service=nginx-web-server" -o result\_requests.json )

**فاز پیش بینی تعداد فراخوانی ها در زمان های بعدی**

در این فاز به دنبال پیش بینی تعداد فراخونی در زمان های بعدی با به کار گیریgraph convolution neural network هستیم .

برای پیش بینی تعداد فراخوانی ها در زمان های بعدی چون ما با داده های سری زمانی سروکار داریم و نیاز داریم از recurrent neural network ها که مناسب برای این کار هستند استفاده کنیم ، برای پیش بینی با استفاده از دیتاستی که در فاز قبلی به دست آورده ایم استفاده می کنیم . کد

Forcasting\_NumberCall\_for\_Microservices برای پیش بینی استفاده شده است ،این کد با تعداد epoch حدودا 500 ، به این متریک ها رسیده است :

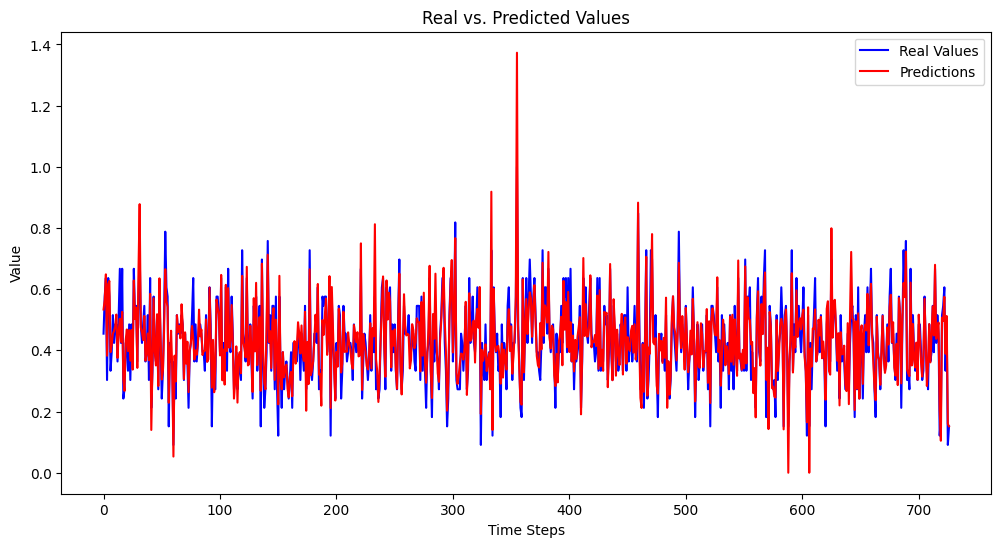
**model.compile(optimizer='adam',loss='mean\_squared\_error',metrics=['mae','mape'])**

mape: 10.2556

loss: 0.0027

mae: 0.0375

mape: 10.2556



کد تا حد زیادی داکیومنت شده و برای بخش های مختلف کد توضیح آن قطعه کد نوشته شده است .