

شبیه سازی کامپیوتری - پروژه پایانی

پویا یوسفی - ۹۸۱۷۱۲۲۳

امیررضا باقری دلویی - ۹۸۱۰۹۸۰۴

چکیده روند پروژه

در این پروژه هدف آن است که یک سیستم درخواست غذای آنلاین شبیه سازی شود. توابع و کدها در ۴ بخش در ادامه توضیح داده می شوند و پس از آن نتایج و گزارش پروژه ارائه می شود.

۱. کلاس Restaurant

در این کلاس، ورودی های env، num_instances و max_wait_time_list داده می شوند که به ترتیب نمایانگر محیط simpy (Environment)، تعداد سرویس دهنده ها، و ماکزیمم زمان انتظار برای هر درخواست می باشد.

سپس به تعداد هر سرویس دهنده، از آن سرویس دهنده یک PriorityResource ایجاد می شود. تفاوت PriorityResource با Resource عادی، در هندل کردن اولویت درخواست ها است.

بعد از آن، مقادیر میانگین زمان سرویس و ماکزیمم زمان انتظار درخواست ها مقداردهی می شوند. توابع پردازش هر سرویس دهنده نیز در این کلاس تعریف شدند که در آنها، به اندازه عدد رندوم با توزیع نمایی و میانگین زمان سرویس، env.timeout می شود.

۲. سرویس دهنده ها (Servers)

هر سرویس دهنده، ورودی های env و customer و restaurant و priority و request_type را دریافت می کند که به ترتیب نمایانگر مشتری کنونی، رستوران، اولویت درخواست و نوع درخواست، هستند.

ابتدا از طرف سرویس دهنده درخواست (request) ایجاد می شود و مشتری وارد صف می شود. در این مرحله مقدار queue_in که نمایانگر زمان ورود به صف هست برابر با env.now() مقداردهی می شود. در دیکشنری time_queue، برای هر سرویس دهنده زمان ورود به صف ذخیره می شود. در دیکشنری len_queue برای هر سرویس دهنده طول صف در هنگام ورود به صف و خروج از صف ذخیره می شود.

بعد از آنکه درخواست (request) تمام شد، مشتری از صف خارج می شود و فرایند سرویس دهی به او شروع می شود. زمان خروج از صف مجدداً با دستور env.now اندازه گیری می شود و مقادیر حضور در صف (time_in_queue) به دیکشنری مربوط به آن اضافه می شوند. در انتها با دستور env.process سرویس دهی انجام می شود.

۳. توابع انجام درخواست

برای هر نوع درخواست، بسته به آنکه از چه سرویس دهنده‌هایی قرار است استفاده کند، توابعی پیاده سازی شدند. ورودی‌های این توابع env و customer و restaurant هستند. مقدار اولویت (Priority) درخواست تعیین می‌شود و به ترتیب با دستور env.process سرویس دهی به او انجام می‌شود.

۴. اجرای شبیه سازی

برای اجرای شبیه سازی، پس از آنکه ورودی‌های اولیه از کاربر دریافت شدند (یا به صورت پیش فرض تعیین شدند) تابع run_restaurant صدا زده می‌شود. ابتدا یک نمونه از کلاس Restaurant ایجاد می‌شود و در یک حلقه بی‌نهایت، مشتری‌ها پس از گذشتن زمان arrival_rate اضافه می‌شوند.

سپس تابع env.process روی تابع go_to_restaurant اجرا می‌شود. در تابع go_to_restaurant ابتدا یک عدد رندوم بین ۱ تا ۱۰۰ انتخاب می‌شود و با توجه به توزیع احتمالی تجمعی (CDF) با توجه به احتمال هر درخواست که در داک پروژه آورده شده بود، تصمیم گرفته می‌شود که نوع درخواست چگونه باشد و آن درخواست اجرا می‌شود.

دیکشنری‌های in_queue و time_queue و len_queue که به ترتیب نشان دهنده زمان حضور در صف، زمان ورود و خروج از صف، و تعداد مشتریان در صف، برای هر سرویس دهنده هستند، ایجاد می‌شوند. دیکشنری in_queue_request دیکشنری‌ای است که مقادیر حضور در صف را برای هر درخواست نشان می‌دهد.

برای بدست آوردن خروجی‌های خواسته شده، مراحل زیر را طی می‌کنیم:

ابتدا با استفاده از تابع get_df_server، برای هر سرویس دهنده یک دیتافریم تشکیل می‌دهیم که ستون‌های آن نشان دهنده زمان ورود و خروج مشتری به صف‌های آن سرویس دهنده‌ها و طول صف در آن لحظات است.

سپس با استفاده از تابع get_avg_queue، بازه‌های زمانی در صف‌های هر سرویس دهنده را بدست می‌آوریم و براساس آن بازه‌های زمانی، میانگین وزن دار طول صف را حساب می‌کنیم.

برای بدست آوردن بهره‌وری هر سرویس دهنده، تابع get_server_utilization تعریف شده است. در این تابع ابتدا زمان آزاد هر سرویس دهنده را با استفاده از زمان‌هایی که طول صف سرویس دهنده برابر با صفر بوده است، بدست می‌آوریم. همچنین ممکن است سرویس دهنده تا زمان پایان شبیه سازی بیکار باشد، بنابراین تفاوت زمان آخرین سرویس او و زمان شبیه سازی را نیز به زمان آزاد اضافه می‌کنیم. همچنین زمان ۰ تا شروع اولین سرویس آن را

نیز به زمان آزاد اضافه می‌کنیم. نهایتاً زمان آزاد را تقسیم به زمان کل شبیه سازی کرده و آن را از ۱ کم می‌کنیم و ضربدر ۱۰۰ می‌کنیم تا درصد بهره‌وری بدست بیاید.

نتایج بدست آمده برای هر سرویس‌دهنده در یک دیتافریم برای نمایش تجمیع شدند.

برای هر درخواست نیز، میانگین زمان انتظار را با استفاده از دیکشنری `in_queue_request` بدست می‌آوریم، و در یک دیتافریم نمایش می‌دهیم.


۵. نتایج

نتایج شبیه سازی با ورودی‌های زیر، در ادامه آورده می‌شود:

```
# test 1
num_instances = [1, 1, 1, 2, 5, 3, 2]
arrival_rate = 1/30
simulation_time = 1000
max_wait_time_list = [25, 30, 25, 30, 30, 40, 20]
```

```
[49] pd.DataFrame({"Average Queue Length": list(avg_queue_len.values()),
                  "Average Waiting Time": list(avg_delay.values()),
                  "Server Utilization": list(server_utilization.values())}, avg_queue_len.keys())
```

	Average Queue Length	Average Waiting Time	Server Utilization
Api Port	7632.161975	379.063213	99.97
Delivery Server	0.211647	1.656025	15.14
Order Manager	773.232410	423.172345	99.97
Pay Server	0.008668	0.050647	0.83
Restaurant Manager	158.978391	378.955165	99.74
Web Port	6278.646147	450.280570	99.99

Average Waiting Time 	
Check Info Phone	0.001492
Check Info Web	0.000000
Order Phone	352.944197
Order Web	401.580461
Request Delivery	358.524436
Track Order	0.000000

```
[52] num_requests = sum([len(requests) for requests in in_queue_request.values()])
total_avg_waiting_time = sum([sum(value) for value in in_queue_request.values()])/num_requests
print(f"Total Number of Requests: {num_requests} with Average of {total_avg_waiting_time} Waiting Time")
```

Total Number of Requests: 2768 with Average of 358.4794641405608 Waiting Time

در نتایج دیده می‌شود که بیشترین طول صف مربوط به Port های API و Web بوده است که با توجه به اینکه همه درخواست‌ها از این دو درگاه می‌گذرند، منطقی است. بیشترین زمان‌های انتظار در صف مربوط به سرویس‌دهنده‌های درگاه موبایل، درگاه وب، مدیر رستوران، و مدیر سفارش است. علت آن هم به دلیل این است که این سرویس‌دهنده‌ها تقریباً در تمامی دستورات مشترک هستند و همچنین احتمال وقوع آنها بالا است. در مقابل، ارتباط با پیک، و پرداخت، زمان در انتظار بسیار کمی دارند.

همچنین برای میانگین زمان انتظار هر درخواست، مشاهده می‌شود که بیشترین زمان انتظار مربوط به سفارش با موبایل یا وب است و همچنین درخواست پیک. این درخواست‌ها با احتمال بالایی به وقوع می‌پیوندند و بنابراین زمان در انتظار بیشتری دارند.

در دو حالت دیگر نتایج را مقایسه می‌کنیم:

```
# test 2
num_instances = [2, 3, 1, 1, 1, 6, 6]
arrival_rate = 1/30
simulation_time = 1000
max_wait_time_list = [25, 30, 25, 30, 30, 40, 20]
```

در این تست، تعداد درگاه‌های موبایل، درگاه‌های وب، مدیریت رستوران‌ها و مدیریت مشتریان، افزایش یافتند اما تعداد سرویس‌دهنده‌های پرداخت‌ها و ارتباط با پیک کاهش پیدا کردند. نتایج این شبیه سازی را در شکل زیر مشاهده می‌کنید:

	Average Queue Length	Average Waiting Time	Server Utilization
Api Port	6817.202889	248.097564	99.96
Web Port	5618.488884	373.564948	99.94
Order Manager	1774.136705	460.096287	99.93
Pay Server	39.762834	215.329722	99.46
Customer Manager	0.000000	0.000000	0.00
Delivery Server	72.896259	275.597388	99.48
Restaurant Manager	568.687493	429.993876	99.91

	Average Waiting Time
Order Phone	258.039401
Order Web	377.910388
Delivery Message	0.000000
Check Info Phone	0.097037
Check Info Web	0.000000
Request Delivery	373.249014
Track Order	0.160052

```
[31] num_requests = sum([len(requests) for requests in in_queue_request.values()])
total_avg_waiting_time = sum([sum(value) for value in in_queue_request.values()])/num_requests
print(f"Total Number of Requests: {num_requests} with Average of {total_avg_waiting_time} Waiting Time")
```

Total Number of Requests: 5637 with Average of 308.2377434645902 Waiting Time

مشاهده می‌شود که مقداری از زمان انتظار درگاه‌های موبایل و وب کاهش یافته است، اما در عوض زمان انتظار سرویس‌دهنده‌های پرداخت‌ها و ارتباط با پیک، افزایش پیدا کرده است.

زمان انتظار درخواست‌ها تغییر چندانی نکرده است و همچنان بیشترین زمان انتظار مربوط به درخواست‌های سفارش با موبایل و وب و درخواست پیک است.

همچنین مشخص است که با افزایش تعداد سرویس‌دهنده‌ها، تعداد درخواست‌های بیشتری انجام شده‌اند و میانگین کلی زمان انتظار در سیستم کاهش یافته است.

تست ۳:

```
# test 3
num_instances = [4, 1, 4, 4, 4, 10, 10]
arrival_rate = 1/30
simulation_time = 1000
max_wait_time_list = [25, 30, 25, 30, 30, 40, 20]
```

در این حالت، تعداد سرویس‌دهنده‌ها را (به جز مدیر مشتری) افزایش دادیم. تعداد درگاه‌ها بیشتر از بقیه افزایش یافتند.

	Average Queue Length	Average Waiting Time	Server Utilization
Api Port	5770.898675	78.206981	99.91
Web Port	5030.369373	313.993701	99.94
Order Manager	2734.088402	452.277308	99.78
Pay Server	140.306112	231.347224	98.44
Customer Manager	0.000000	0.000000	0.00
Delivery Server	9.228483	19.932225	94.09
Restaurant Manager	896.723267	383.358835	99.88

	Average Waiting Time
Order Phone	120.647278
Order Web	320.748828
Delivery Message	0.000000
Check Info Phone	0.007252
Check Info Web	0.014441
Request Delivery	283.149175
Track Order	0.000000

```
[49] num_requests = sum([len(requests) for requests in in_queue_request.values()])
total_avg_waiting_time = sum([sum(value) for value in in_queue_request.values()])/num_requests
print(f"Total Number of Requests: {num_requests} with Average of {total_avg_waiting_time} Waiting Time")
```

Total Number of Requests: 10345 with Average of 195.20523904013643 Waiting Time

مشاهده می‌شود که افزایش سرویس‌دهنده‌ها باعث کاهش طول صف و همچنین زمان انتظار می‌شود، اما همچنان این مقادیر بالا هستند. زمان انتظار کلی سیستم به میزان زیادی کاهش یافته است و نسبت به تست ۱، به نصف رسیده است.

پیشنهاد برای بهبود: مشخص است که سرویس‌دهنده‌ها کم هستند و این مسئله از میزان بهره‌وری آنها و زمان انتظار برای آنها مشخص است. پیشنهاد آن است که تعداد سرویس‌دهنده‌های درگاه موبایل، درگاه وب، مدیر سفارش، پرداخت، و مدیر رستوران افزایش قابل توجهی کنند. مثلاً مثال زیر را در نظر بگیرید:

Improvement

```
num_instances = [30, 10, 30, 30, 30, 30, 30]
arrival_rate = 1/30
simulation_time = 1000
max_wait_time_list = [25, 30, 25, 30, 30, 40, 20]
```

	Average Queue Length	Average Waiting Time	Server Utilization
Api Port	868.554336	49.160177	99.60
Web Port	1684.261087	40.034030	99.42
Order Manager	2751.333593	226.082159	99.47
Pay Server	1181.597320	232.411595	98.38
Customer Manager	0.242314	0.192928	8.13
Delivery Server	820.742160	60.233669	97.97
Restaurant Manager	4644.731893	188.307207	99.65

	Average Waiting Time
Order Phone	103.796069
Order Web	105.246588
Delivery Message	40.931966
Check Info Phone	81.587477
Check Info Web	410.425440
Request Delivery	69.692326
Track Order	81.922458

```
[90] num_requests = sum([len(requests) for requests in in_queue_request.values()])
total_avg_waiting_time = sum([sum(value) for value in in_queue_request.values()])/num_requests
print(f"Total Number of Requests: {num_requests} with Average of {total_avg_waiting_time} Waiting Time")
```

Total Number of Requests: 40747 with Average of 92.16793384867321 Waiting Time

مشاهده می‌شود که مقادیر زمان انتظار کاهش قابل توجهی کرده‌اند و همچنین تعداد درخواست‌هایی که انجام می‌شود نیز افزایش قابل توجهی داشته است. بنابراین با افزایش سرویس‌دهنده‌ها خروجی‌های سیستم مطلوب‌تر می‌شوند.

تغییر معماری سیستم: مشاهده می‌شود که درگاه‌ها بسیار شلوغ بودند و پس از آن این شلوغی به بخش مدیریت رستوران، مدیریت مشتریان و مدیریت سفارش‌ها منتقل می‌شود. پیشنهاد می‌شود که برای بعضی از درخواست‌ها که لزومی به دخالت مدیریت نیست، مستقیم از درگاه به پیک یا بخش پرداخت متصل شود. یا آنکه تقسیم بندی درخواست‌ها بین سرویس‌دهنده‌ها تغییر کند. برای مثال مدیریت مشتری بسیار بیکار بوده است در صورتی که میشد بعضی از درخواست‌ها از مسیر مدیریت مشتری انجام شود.