شبیه سازی کامپیوتری

پروژه، فاز دوم

دکتر صفایی

مهران بختیاری

کد کامل شبیه‌سازی:

کد کامل شبیه‌سازی در فایل main.py آمده است. در بخش اول کد، می‌توانیم مقادیر پارامترهای سیستم را تغییر دهیم و همچنین از بین سه حالت مختلف برای قانون صف، یکی را انتخاب کنیم.

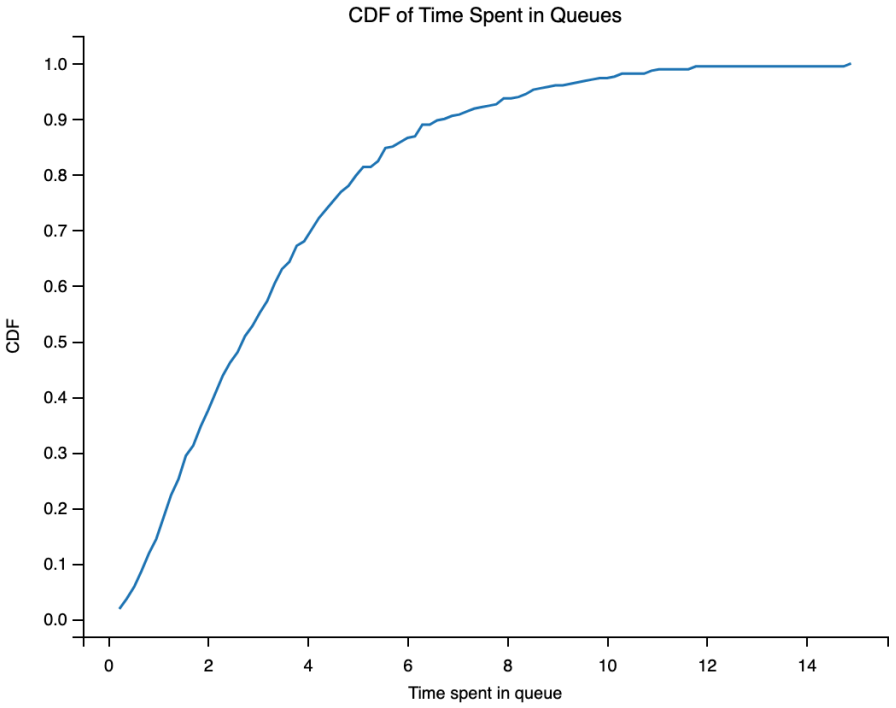
نتایج و خواسته‌ها:

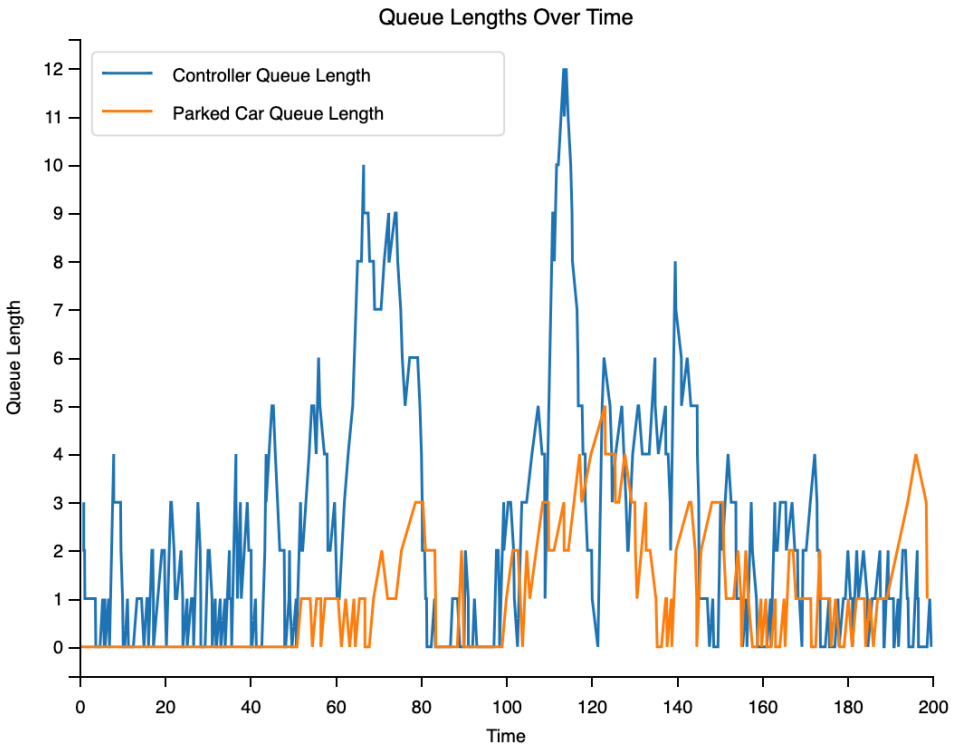
میانگین طول صف‌ها، میانگین زمان صرف شده در صف‌ها، میانگین بهره‌وری هر کدام از پردازنده‌ها در کنترلر و ماشین پارک شده، توان عملیاتی هر کدام از پردازنده‌ها در کنترلر و ماشین پارک شده، نمودار CDF مربوط به مدت‌زمان صرف شده در تمام صف‌ها برای انواع تسک‌ها، نمودار طول صف‌ها در طول زمان، نمودار فرکانس زمان‌های صرف شده در صف ها:

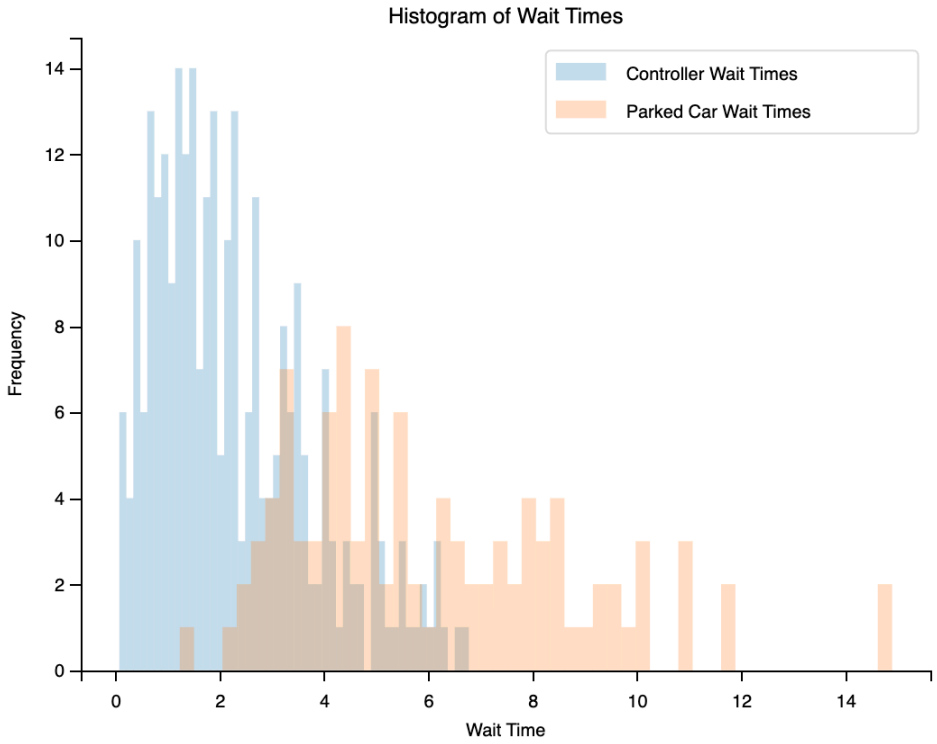
```
Lambda1 = 1.5
Lambda2 = 1.0
X = 0.5
C = 1.0
t = 50
T = 200
N = 2
P = 0.5

queuing_mode = 'FIFO'
```

Average queue length (Controller): 2.3367875647668392
Average queue length (Parked Car): 0.9455445544554455
Average wait time (Controller): 2.2906428296466728
Average wait time (Parked Car): 6.054479889152932
Efficiency (Controller Processor 0): 0.4833845867569373
Throughput (Controller Processor 0): 0.725
Efficiency (Controller Processor 1): 0.45511495168648536
Throughput (Controller Processor 1): 0.67
Efficiency (Parked Car): 0.680456809491743
Throughput (Parked Car): 0.6866666666666666


CDF of Time Spent in Queues
CDF
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
2
4
6
8
10
12
14
Time spent in queue

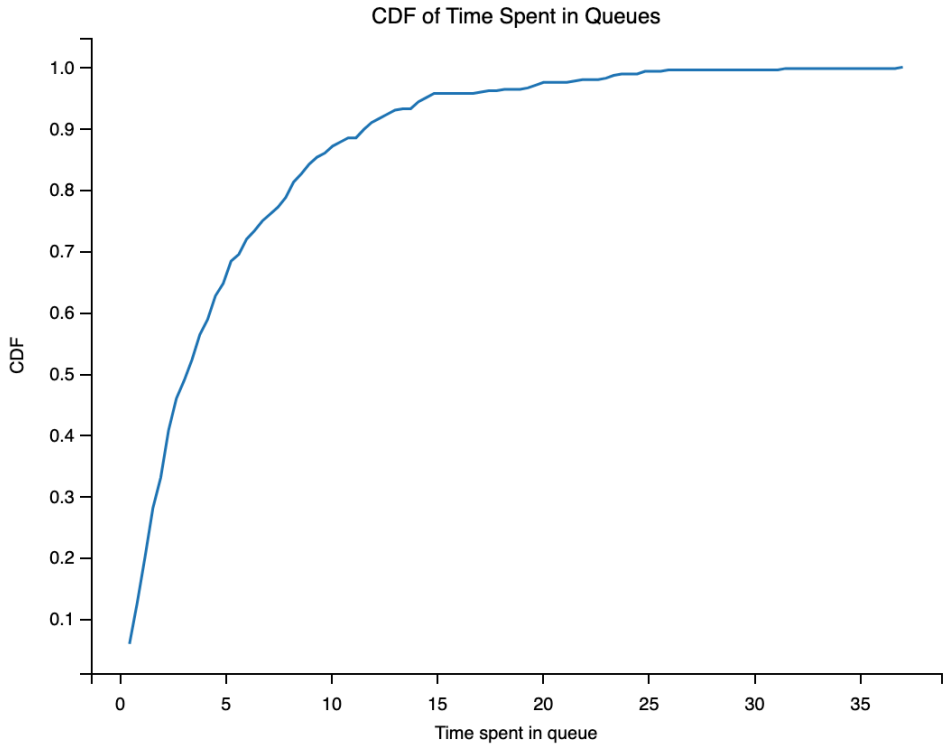

Queue Lengths Over Time
Controller Queue Length
Parked Car Queue Length
Queue Length
12
11
10
9
8
7
6
5
4
3
2
1
0
0
20
40
60
80
100
120
140
160
180
200
Time

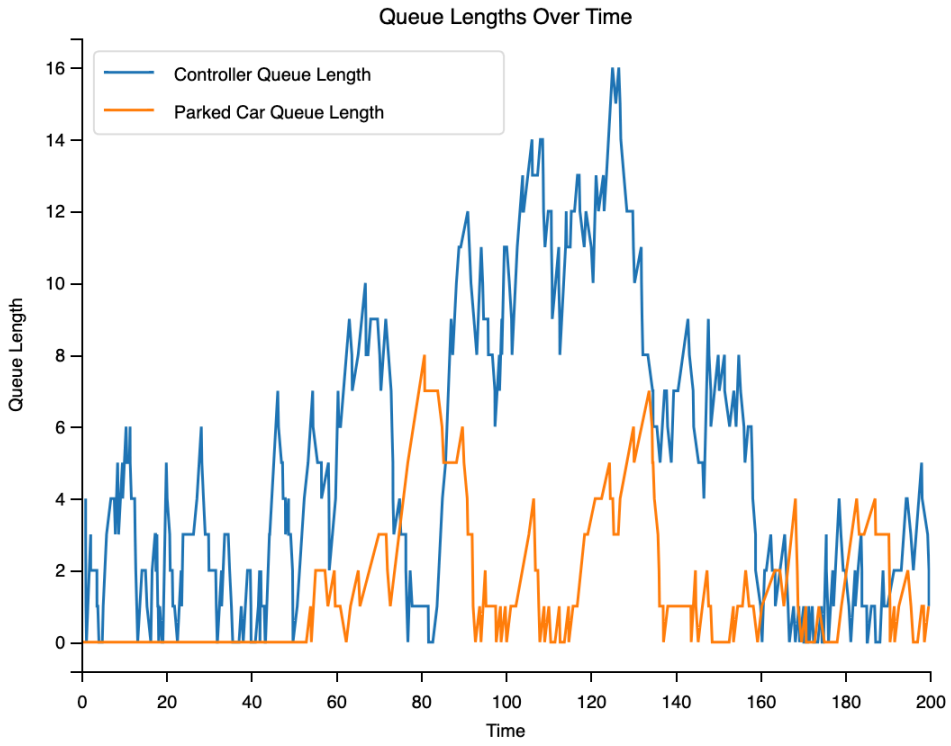


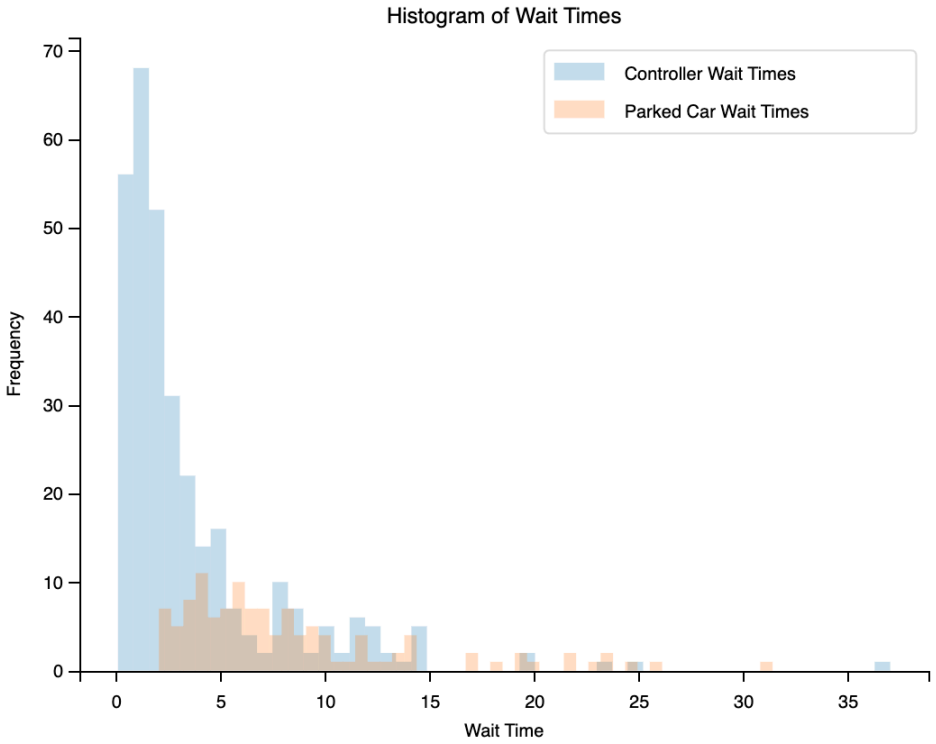

```
Lambda1 = 1.5
Lambda2 = 1.0
X = 0.5
C = 1.0
t = 50
T = 200
N = 2
P = 0.5

queuing_mode = 'WRR'
```

Average queue length (Controller): 4.791891891891892
Average queue length (Parked Car): 1.3726415094339623
Average wait time (Controller): 3.667910671909835
Average wait time (Parked Car): 8.456359689404234
Efficiency (Controller Processor 0): 0.5822378182308414
Throughput (Controller Processor 0): 0.8
Efficiency (Controller Processor 1): 0.570846260341261
Throughput (Controller Processor 1): 0.82
Efficiency (Parked Car): 0.7266869977454801
Throughput (Parked Car): 0.8


CDF of Time Spent in Queues
CDF
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
5
10
15
20
25
30
35
Time spent in queue

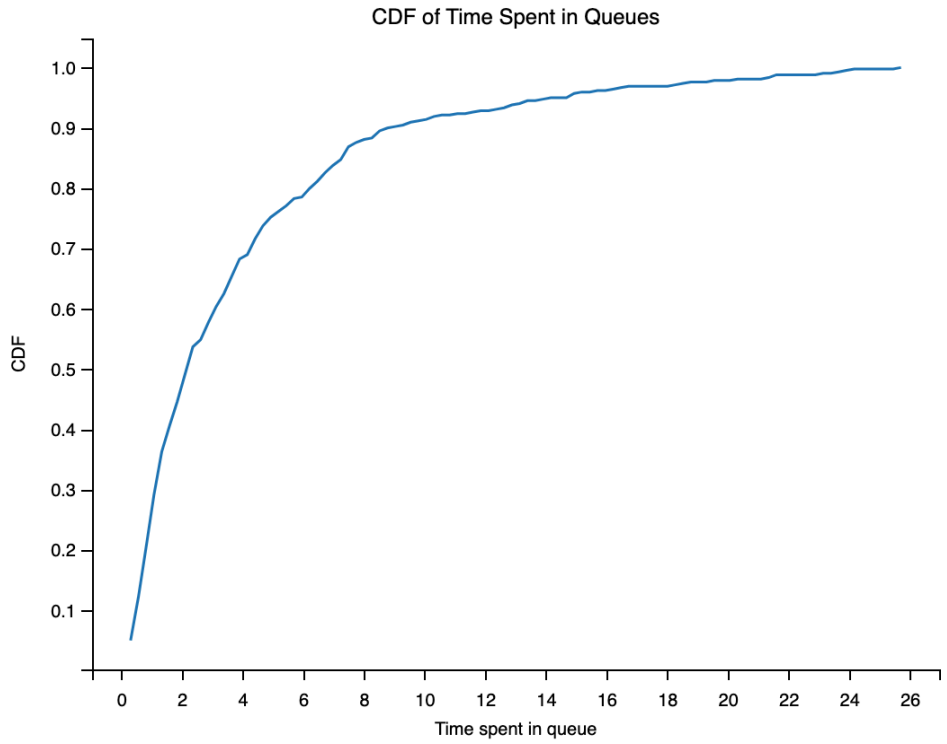

Queue Lengths Over Time
Controller Queue Length
Parked Car Queue Length
Queue Length
16
14
12
10
8
6
4
2
0
0
20
40
60
80
100
120
140
160
180
200
Time

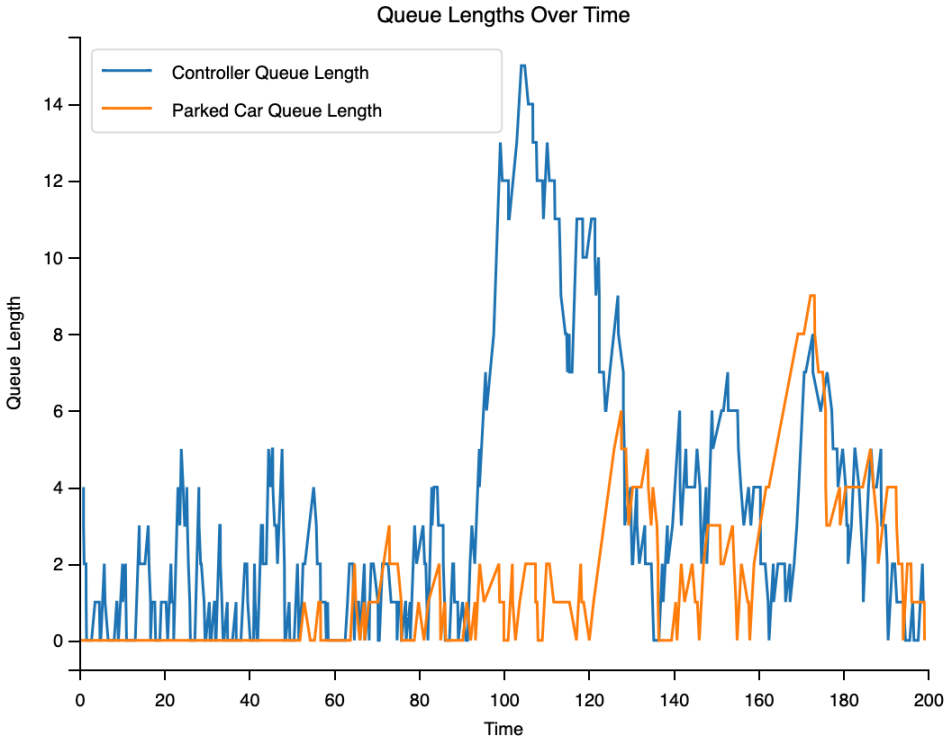


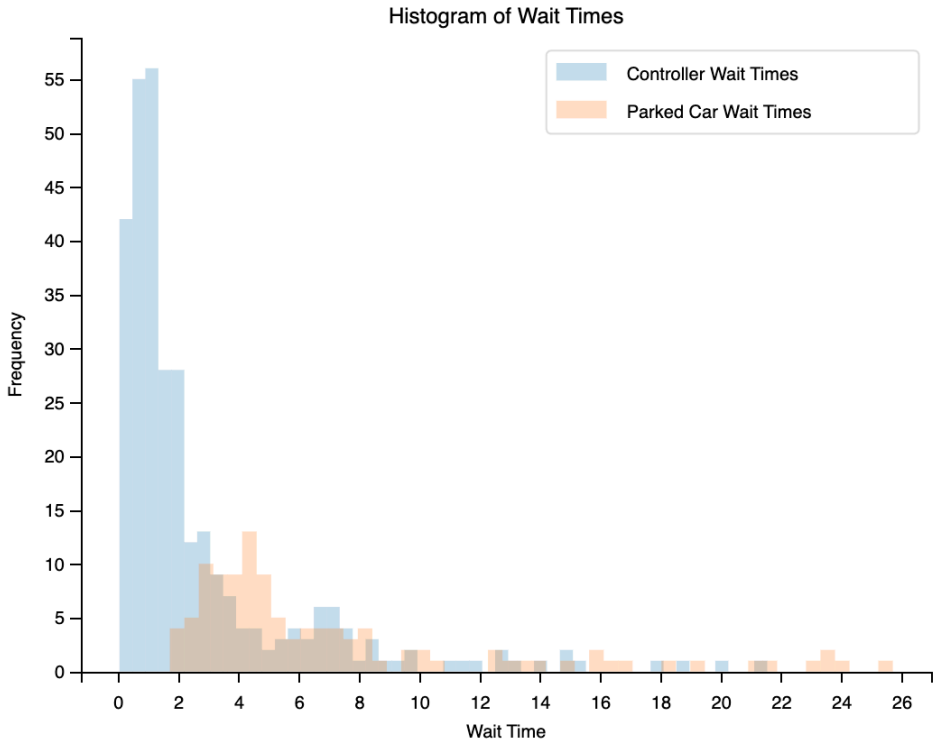

```
Lambda1 = 1.5
Lambda2 = 1.0
X = 0.5
C = 1.0
t = 50
T = 200
N = 2
P = 0.5

queuing_mode = 'NPPS'
```

Average queue length (Controller): 3.092269326683292
Average queue length (Parked Car): 1.4320388349514563
Average wait time (Controller): 2.707434831790144
Average wait time (Parked Car): 7.200562064615632
Efficiency (Controller Processor 0): 0.5017177029187105
Throughput (Controller Processor 0): 0.76
Efficiency (Controller Processor 1): 0.4705190951881194
Throughput (Controller Processor 1): 0.785
Efficiency (Parked Car): 0.7011531840533631
Throughput (Parked Car): 0.74


CDF of Time Spent in Queues
CDF
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
2
4
6
8
10
12
14
16
18
20
22
24
26
Time spent in queue


Queue Lengths Over Time
Controller Queue Length
Parked Car Queue Length
Queue Length
14
12
10
8
6
4
2
0
0
20
40
60
80
100
120
140
160
180
200
Time



راهکارهایی برای افزایش بهره‌وری سیستم:

تنظیم سیاست های زمان‌بندی تسک‌ها:
تنظیم اولویت پویا: اولویت تسک‌ها را به صورت پویا و بر اساس طول صف‌ها و لود سیستم تنظیم کنیم تا از bottleneckها جلوگیری شود.
سیاست‌های صف ترکیبی: چندین سیاست صف را با هم ترکیب کنیم (مثلاً در ابتدا از NPPS استفاده کنیم و در زمان لود بالا به FIFO سوییچ کنیم) تا لود و بهره‌وری پردازش متعادل شود.

افزایش تعداد پردازنده‌ها در کنترلر:
تعداد پردازنده ها در کنترلر (N) را افزایش دهیم تا تسک‌های بیشتری را به صورت همزمان انجام دهیم. این کار می تواند میانگین زمان صرف شده در صف‌ها و میانگین طول صف‌ها را کاهش دهد.

بهینه سازی اورهد:
اورهد مربوط به انتقال تسک‌ها به ماشین پارک شده (C) را کاهش دهیم. این کار را می توانیم از طریق به کار بردن مکانیسم های انتقال تسک کارآمدتر انجام دهیم.

پردازش دسته ای تسک‌ها:
در صورت امکان، وظایف را به صورت دسته ای پردازش کنیم، این کار می‌تواند اورهد context switching را کاهش دهد و توان عملیاتی (throughput) را بهبود بخشد.

تعادل لود:
استراتژی های متعادل کننده لود را پیاده سازی کنیم تا تسک‌ها به طور مساوی در بین پردازنده های در دسترس توزیع شوند، به عبارتی اطمینان حاصل کنیم که هیچ پردازنده ای لود بیش از اندازه نداشته باشد، در حالی که سایر پردازنده‌ها بیکار و در حالت idle هستند.

بهبود کنترل تولید تسک‌ها:
نرخ تولید تسک‌ها (X) را کنترل کنیم تا مطمئن شویم که سیستم با تسک‌های بیش از اندازه در یک زمان overwhelme نمی‌شود.

افزایش تخصیص منابع:
منابع محاسباتی مانند حافظه و قدرت پردازشی را که برای کنترلر و ماشین پارک شده در دسترس است افزایش دهیم تا تسک‌ها بهتر و سریع‌تر انجام شوند.

بهینه سازی احتمال P:
احتمال P را برای ارسال تسک‌ها به ماشین پارک شده بهینه کنیم. در واقع حالت تعادلی را بیابیم که در آن تسک‌های کافی برای خودروی پارک شده ارسال می شود، بدون اینکه آن را تحت فشار قرار دهیم یا از آن کم استفاده کنیم.

پیاده سازی الگوریتم های پیش بینی کننده:
از الگوریتم های پیش بینی برای پیش بینی لود تسک‌ها و تنظیم ظرفیت پردازش به صورت پویا استفاده کنیم. برای مثال، مدل‌های یادگیری ماشینی می‌توانند بازه‌های زمانی شلوغ را پیش‌بینی کنند و منابع بیشتری را در آن زمان‌ها تخصیص دهند.

سیاست های مختلف برای صف‌ها را مقایسه کنید و بیان کنید کدام سیاست بهتر است و چرا؟

سیاست FIFO یا First In First Out:
تسک‌ها به ترتیبی که می رسند پردازش می شوند. این سیاست برای پیاده سازی و درک ساده است و برای تسک‌ها منصفانه است، زیرا آن‌ها به ترتیب ورود پردازش می شوند. در عین حال، در صورتی که تسک‌های با اولویت بالا در پشت تسک‌های با اولویت پایین گیر کنند، می تواند منجر به ناکارآمدی شود. بهترین زمان استفاده و حالت بهینه برای استفاده از این سیاست، در سناریوهایی است که همه تسک‌ها دارای اولویت و زمان پردازش مشابه هستند.

سیاست WRR یا Weighted Round Robin:
بر اساس اولویت هر تسک، یک وزن به آن نسبت داده می‌شود و تسک‌ها با اولویت بالاتر بیشتر پردازش می شوند. این سیاست تضمین می کند که تسک‌ها با اولویت بالا بیشتر پردازش می شوند و همچنین با دادن سهم عادلانه از زمان پردازش به همه تسک‌ها، لود سیستم را متعادل می کند. در عین حال، پیاده سازی آن در مقایسه با FIFO پیچیده تر است و اگر توزیع وزن‌ها بهینه نباشد، باز هم می تواند منجر به تأخیر در پردازش تسک‌ها با اولویت بالا شود. استفاده از این سیاست در سناریوهایی خوب است که تسک‌ها دارای اولویت‌های متفاوتی هستند و می‌خواهیم از پردازش منصفانه و در عین حال اولویت دادن به تسک‌ها با اولویت بالا اطمینان حاصل کنیم.

سیاست NPPS یا Non Preemptive Priority Scheduling:
تسک‌ها دقیقاً و به طور سخت‌گیرانه ای، بر اساس اولویتشان پردازش می شوند و تسک‌ها با اولویت بالاتر همیشه در ابتدا پردازش می شوند. این سیاست تضمین می کند که تسک‌ها با اولویت بالا در اسرع وقت پردازش می‌شوند و می

تواند زمان انتظار برای تسک‌های مهم را به میزان قابل توجهی کاهش دهد. در عین حال، تسک‌هایی که اولویت کمتری دارند ممکن است دچار starvation شوند و در واقع هرگز پردازش نشوند، همچنین اگر تسک‌ها با اولویت بالا نرخ کمی داشته باشند، می تواند منجر به ناکارآمدی شود. استفاده از این سیاست برای سناریوهایی ایده آل است که تسک‌های خاصی باید فوراً پردازش شوند و نمی‌توانند پشت تسک‌های با اولویت پایین‌تر منتظر بمانند.

بنابراین در مجموع می‌توان گفت FIFO ساده ترین برای پیاده سازی است، اما اگر اولویت های تسک‌ها به طور قابل توجهی متفاوت باشد، ممکن است کارآمدترین نباشد. پیاده سازی WRR به دلیل محاسبات وزن های نسبت داده شده پیچیده تر است و با اطمینان حاصل کردن از پردازش منصفانه تسک‌ها و در عین حال اولویت دادن به تسک‌های مهم تر، تعادلی را ارائه می دهد. NPPS به مرتب‌سازی تسک‌ها بر اساس اولویت نیاز دارد و پیاده سازی آن کمی پیچیدگی را اضافه می‌کند، برای تسک‌های با اولویت بالا بسیار کارآمد است اما می تواند منجر به عدم پردازش تسک‌ها با اولویت پایین تر شود. بنابراین به طور کلی، WRR بهترین سیاست است، اما بر اساس نتایج شبیه‌سازی های انجام شده که در بالا آوردیم و به ازای مقادیر fix شده گفته شده برای پارامترهای سیستم، FIFO عملکرد بهتری داشته است، زیرا مشاهده می‌کنیم که میانگین طول صف‌ها و میانگین زمان صرف شده در صف‌ها برای آن در این شبیه‌سازی از بقیه سیاست ها کمتر است.

تجزیه و تحلیل تاثیر پارامترها:

پارامتر توزیع نمایی (λ): با در نظر گرفتن یک سیاست مشخص برای صف و fix کردن سایر پارامترهای سیستم و تغییر λ (انجام شبیه سازی با λ های مختلف)، می‌بینیم که با افزایش این پارامتر، میانگین زمان انجام تسک‌ها کاهش می‌یابد و به عبارتی میانگین سرعت انجام تسک‌ها افزایش می‌یابد، بنابراین به طور میانگین، تسک‌های بیشتری در یک بازه زمانی مشخص انجام می‌شوند، با پردازش سریع‌تر تسک‌ها، تسک‌ها با سرعت بیشتری از صف خارج می‌شوند، بنابراین میانگین طول صف‌ها کمتر می‌شود، در عین حال توان عملیاتی یا throughput سیستم بهبود پیدا می‌کند و همچنین میانگین زمانی که یک تسک در سیستم سپری می‌کند (شامل زمان منتظر ماندن در صف و زمان پردازش) کاهش می‌یابد.

شبیه‌سازی های انجام شده:

```
Lambda1 = 1.0
Lambda2 = 1.0
X = 0.5
C = 1.0
t = 50
T = 200
N = 2
P = 0.5

queuing_mode = 'FIFO'
```

Average queue length (Controller): 20.163987138263664
Average queue length (Parked Car): 0.5345622119815668
Average wait time (Controller): 14.068298111654629
Average wait time (Parked Car): 23.279727889334886
Efficiency (Controller Processor 0): 0.651710846858641
Throughput (Controller Processor 0): 0.585
Efficiency (Controller Processor 1): 0.6796126123922146
Throughput (Controller Processor 1): 0.745
Efficiency (Parked Car): 0.4942404936756992
Throughput (Parked Car): 0.5666666666666667

```
Lambda1 = 1.5
Lambda2 = 1.0
X = 0.5
C = 1.0
t = 50
T = 200
N = 2
P = 0.5

queuing_mode = 'FIFO'
```

Average queue length (Controller): 3.25
Average queue length (Parked Car): 1.5432692307692308
Average wait time (Controller): 2.8041462787554616
Average wait time (Parked Car): 7.796285748955593
Efficiency (Controller Processor 0): 0.4890982389934361
Throughput (Controller Processor 0): 0.705
Efficiency (Controller Processor 1): 0.4298219350239127
Throughput (Controller Processor 1): 0.76
Efficiency (Parked Car): 0.7063520843885325
Throughput (Parked Car): 0.7533333333333333

```
Lambda1 = 2.0
Lambda2 = 1.0
X = 0.5
C = 1.0
t = 50
T = 200
N = 2
P = 0.5

queuing_mode = 'FIFO'
```

Average queue length (Controller): 1.9932885906040267
Average queue length (Parked Car): 1.2819148936170213
Average wait time (Controller): 1.6722482509895216
Average wait time (Parked Car): 6.119590722134097
Efficiency (Controller Processor 0): 0.41992379874389585
Throughput (Controller Processor 0): 0.78
Efficiency (Controller Processor 1): 0.36844425578590545
Throughput (Controller Processor 1): 0.795
Efficiency (Parked Car): 0.8147298874109109
Throughput (Parked Car): 0.7066666666666667

پارامتر اورهد (C): با در نظر گرفتن یک سیاست مشخص برای صف و fix کردن سایر پارامترهای سیستم و تغییر C (انجام شبیه سازی با C های مختلف)، می‌بینیم که با افزایش این پارامتر، زمان انتقال تسک‌ها به ماشین پارک شده افزایش می‌یابد و زمان بیشتری طول می‌کشد تا تسک‌هایی که به ماشین پارک شده منتقل شده اند، انجام شوند. اگر این اورهد زیاد باشد، تسک های کمتری به ماشین پارک شده فرستاده می‌شوند که این موضوع منجر به افزایش طول صف کنترل می‌شود، همچنین کارایی سیستم ممکن است به دلیل افزایش زمان منتقل کردن تسک‌ها به جای پردازش آن‌ها، کاهش پیدا کند و میانگین زمانی که یک تسک در سیستم سپری می‌کند (مخصوصا برای تسک‌هایی که به ماشین پارک شده فرستاده می‌شوند) افزایش می‌یابد.

شبیه‌سازی های انجام شده:

```
Lambda1 = 1.5
Lambda2 = 1.0
X = 0.5
C = 0.5
t = 50
T = 200
N = 2
P = 0.5

queuing_mode = 'FIFO'
```

Average queue length (Controller): 2.2150776053215075
Average queue length (Parked Car): 2.014705882352941
Average wait time (Controller): 1.8422192338415593
Average wait time (Parked Car): 6.83045110381457
Efficiency (Controller Processor 0): 0.5399719513486907
Throughput (Controller Processor 0): 0.89
Efficiency (Controller Processor 1): 0.5027214053233771
Throughput (Controller Processor 1): 0.715
Efficiency (Parked Car): 0.7709517465604585
Throughput (Parked Car): 0.7866666666666666

```
Lambda1 = 1.5
Lambda2 = 1.0
X = 0.5
C = 1.0
t = 50
T = 200
N = 2
P = 0.5

queuing_mode = 'FIFO'
```

Average queue length (Controller): 3.255874673629243
Average queue length (Parked Car): 0.8676470588235294
Average wait time (Controller): 2.7540083603486645
Average wait time (Parked Car): 6.130781417242527
Efficiency (Controller Processor 0): 0.5458163086485619
Throughput (Controller Processor 0): 0.805
Efficiency (Controller Processor 1): 0.545523135219252
Throughput (Controller Processor 1): 0.725
Efficiency (Parked Car): 0.652246693012622
Throughput (Parked Car): 0.66

```
Lambda1 = 1.5
Lambda2 = 1.0
X = 0.5
C = 1.5
t = 50
T = 200
N = 2
P = 0.5

queuing_mode = 'FIFO'
```

Average queue length (Controller): 5.92814371257485
Average queue length (Parked Car): 3.4702380952380953
Average wait time (Controller): 4.944405687266083
Average wait time (Parked Car): 16.271697656563916
Efficiency (Controller Processor 0): 0.48127148043844487
Throughput (Controller Processor 0): 0.745
Efficiency (Controller Processor 1): 0.49608994402944845
Throughput (Controller Processor 1): 0.7
Efficiency (Parked Car): 0.9099267108171102
Throughput (Parked Car): 0.6933333333333334