۱ -

الف) کلاس SRTFQueue را داریم که یک تابع dequeuer دارد. این تابع با پیدا کردن تردی که زمان مورد نیاز برای اتمام‌ش از همه کمتر باشد را خروجی می‌دهد.

خروجی زمان‌بند روی workload3:

0: Arrival of Task 12 (ready queue length = 1)

0: Run Task 12 for duration 2 (ready queue length = 0)

1: Arrival of Task 13 (ready queue length = 1)

2: Arrival of Task 14 (ready queue length = 2)

2: IO wait for Task 12 for duration 1

2: Run Task 14 for duration 1 (ready queue length = 1)

3: Arrival of Task 15 (ready queue length = 2)

3: Wakeup of Task 12 (ready queue length = 3)

3: IO wait for Task 14 for duration 2

3: Run Task 12 for duration 2 (ready queue length = 2)

5: Wakeup of Task 14 (ready queue length = 3)

5: Run Task 14 for duration 1 (ready queue length = 2)

6: Run Task 15 for duration 2 (ready queue length = 1)

8: Run Task 15 for duration 1 (ready queue length = 1)

9: Run Task 13 for duration 2 (ready queue length = 0)

11: Run Task 13 for duration 2 (ready queue length = 0)

13: Run Task 13 for duration 2 (ready queue length = 0)

15: Run Task 13 for duration 1 (ready queue length = 0)

16: Stop

ب)

0: Arrival of Task 12 (ready queue length = 1)

0: Run Task 12 for duration 2 (ready queue length = 0)

1: Arrival of Task 13 (ready queue length = 1)

2: Arrival of Task 14 (ready queue length = 2)

2: IO wait for Task 12 for duration 1

2: Run Task 13 for duration 2 (ready queue length = 1)

3: Arrival of Task 15 (ready queue length = 2)

3: Wakeup of Task 12 (ready queue length = 3)

4: Run Task 14 for duration 1 (ready queue length = 3)

5: IO wait for Task 14 for duration 2

5: Run Task 15 for duration 2 (ready queue length = 2)

7: Wakeup of Task 14 (ready queue length = 3)

7: Run Task 12 for duration 2 (ready queue length = 3)

9: Run Task 14 for duration 1 (ready queue length = 2)

10: Run Task 13 for duration 4 (ready queue length = 1)

14: Run Task 15 for duration 1 (ready queue length = 1)

15: Run Task 13 for duration 1 (ready queue length = 0)

16: Stop

-۲

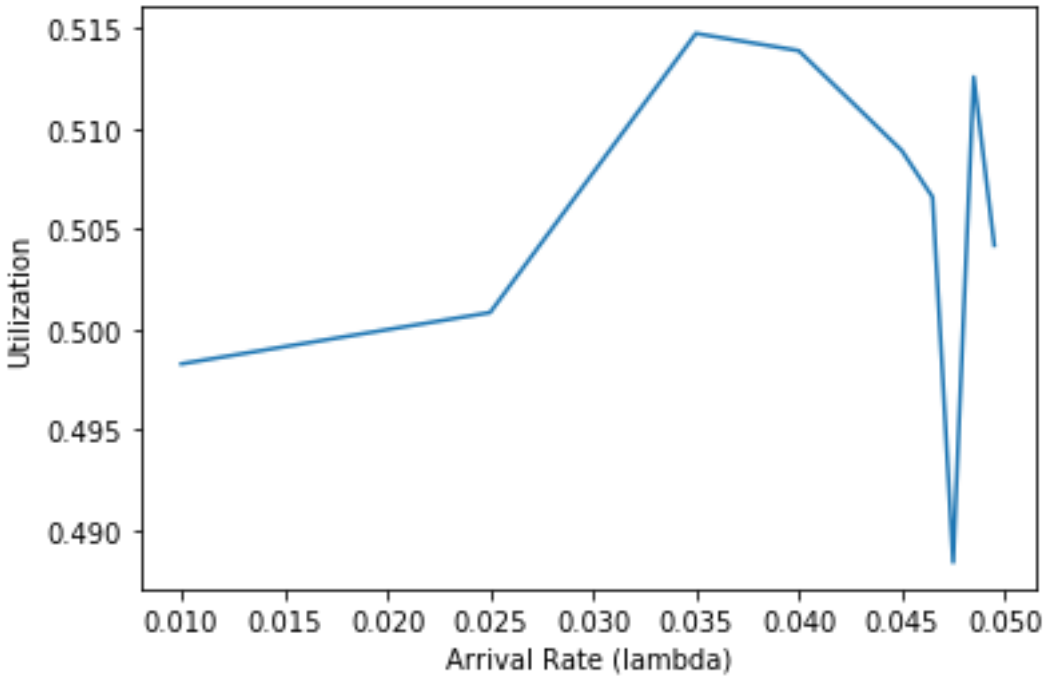
آ) سامانه باز است. در یک سامانه‌ی باز فعالیت‌های جدید مستقل از اتمام فعالیت قبلی می‌رسند. در این سامانه هم با توجه به:

$$ArrivalTime(B_i) = ArrivalTime(B_{i-1}) + X_i$$

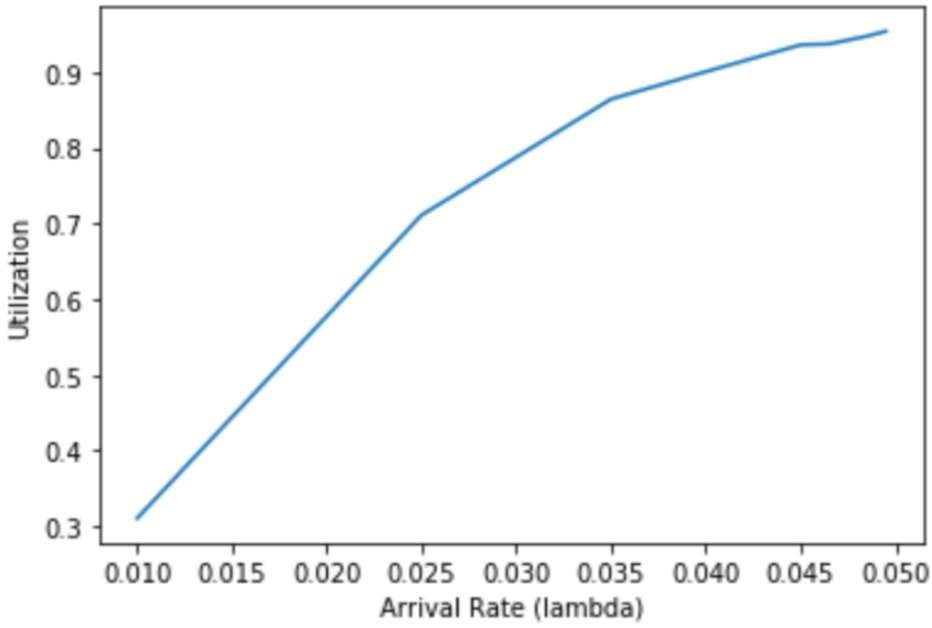
ممکن است هنگام رسیدن فعالیت بعدی، فعالیت قبلی هنوز تمام نشده باشد؛ پس سامانه باز است.

ب) فاصله‌ی زمانی بین ورود دو فعالیت متوالی طبق تعریف از توزیع نمایی تبعیت می‌کند. امیدریاضی متغیر تصادفی نمایی با پارامتر λ برابر $\frac{1}{\lambda}$ است. به همین دلیل λ باید برابر $\frac{1}{M}$ باشد.

ج) نمودار زیر را نگاه کنید. در محور افقی lambda یا پارامتر توزیع نمایی قرار دارد و محور عمودی درصد بهره‌وری است. برای این نمودار از مقدار $M = \frac{0.5}{\lambda}$ برای تمام λ ها استفاده شده که نشان‌دهنده‌ی این است که برای داشتن بهره‌وری ۵۰ درصدی، لازم است تا طول فعالیت یک پردازه نصف میانگین زمان بین ورود دو فعالیت باشد. البته که حدس ما نیز چنین بود و داده‌های بدست آمده از شبیه‌سازی نیز این حدس را تایید می‌کنند.



د) نمودار در دفتر ژوپیتر



ه) بدیهی است که با افزایش λ،زمان بین ورود دو فعالیت که با مقدارλ رابطه عکس دارد کاهش میابد. یعنی پردازنده فرصت کمتری برای استراحت بین فعالیت‌ها دارد و بنابراین بهره‌وری بالاتر است. به همین ترتیب نیز زمان پاسخگویی افزایش میابد زیرا فعالیت ها زودتر ظاهر میشوند اما مقدار کار پردازنده در هر واحد زمانی یکسان است. پس فاصله بیشتری بین ورود یک فعالیت و زمان پایان آن خواهیم داشت و این یعنی زمان پاسخ‌گویی بیشتر.

و) اثبات میشود که srft از نظر مجموع زمان پاسخگویی و کمترین میانه بهترین است اما مثلا صدک ۹۵ ام تاثیر چندانی از srtf نمیگیرد. مثلا اگر round robin داشته باشیم، با هر مقدار quanta ای که شبیه‌سازی کنیم، اعدادی که برای صدک ۹۵ ام بدست می‌آیند دست خوش تغییر چندانی نمیشوند.

ز) البته پاسخ این قسمت بستگی دارد به نحوه مدلسازی ما از ظاهر شدن فعالیت ها. در حقیقت اگر بپذیریم که ظهور فعالیت ها از توزیع پوآسن پیروی میکنند (در نتیجه ظهور یک فعالیت بعد از دیگری مطابق توزیع نمایی است که در صورت سوال داریم) برای افزایش بهره‌وری باید زمان ورود بین فعالیت ها کاهش بیابد و این یعنی افزایش زمان پاسخگویی. زیرا فعالیت ها زود و پشت سر هم ظاهر میشوند ولی دیر به اتمام میرسند. البته این موضوع خیلی بستگی به تعداد فعالیت ها دارد. مثلا اگر فرض کنیم به صورت میانگین هر دو فعالیت با فاصله ۲۰ واحد ظاهر میشوند و طول هر فعالیت ۲۵ واحد باشد، یعنی پردازنده در هر فعالیتی که انجام میدهد در واقع ۵ واحد تاخیر به تاخیر هایش اضافه میشود و این تاخیر در زمان پاسخ گویی فعالیت هایی که دیر تر ظاهر شده‌اند تاثیر بیشتری میگذارد تا فعالیت‌هایی که در ابتدا ظاهر شده اند. این استدلال نیز توسط نمودار تایید میشود. اگر دقت کنیم، شیب نمودار زمان پاسخ‌گویی برای صدک ۹۵ بیشتر از شیب نمودار میانه زمان پاسخ‌گویی است.

آ) چون تنها دو S Task و T را داریم.

ب) هر دوی S_1 و T_1 از یک توزیع می‌آیند و بنابرین $\Pr[S1 < T1] = \Pr[T1 < S1] = 0.5$.

ج) هر یک از S_i ها از توزیع یکسانی می‌آیند. پس بنا بر قضیه حد مرکزی داریم:

$$Z = \lim_{m\to\infty} \sqrt{m}\,\frac{\frac{\sum_{i=1}^{m} S_i}{m} - \mu}{\sigma}$$

که μ برابر میانگین توزیع S_i ها و σ برابر انحراف معیار توزیع S_i ها است. پس برای m های به قدر کافی بزرگ داریم:

$$\frac{CPUTime(S)}{m} = N(\mu, \frac{\sigma^2}{m})$$

در ادامه می‌توان ثابت کرد که اگر میانگین تعدادی متغییر تصادفی iid از توزیع نرمال پیروی کند، توزیع مجموع این متغیرها هم از توزیع نرمال پیروی می‌کند.

$$\bar{Y} = CPUTime(S) = m.\mu$$

$$\sim N\left(m\mu, m^2\frac{\sigma^2}{m}\right) \quad \text{as } n \to \infty \ (n\ is\ constant)$$

$$\sim N(m\mu, m\sigma^2)$$

د) در ادامه از این خاصیت استفاده می‌کنیم که جمع ۲ متغیر نرمال، خود متغیری نرمال خواهد بود. خواهیم داشت:

$$\Pr[\alpha * CPUTime(S) < CPUTime(T)] = \Pr\left[\alpha \frac{CPUTime(S)}{m} < \frac{CPUTime(T)}{m}\right]$$
$$= \Pr\left[N\left(\alpha\mu, \frac{\alpha^2\sigma^2}{m}\right) < N\left(\mu, \frac{\sigma^2}{m}\right)\right]$$
$$= \Pr\left[N\left(\alpha\mu - \mu, \frac{\alpha^2\sigma^2}{m} + \frac{\sigma^2}{m}\right) < 0\right]$$
$$= \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{-(\alpha\mu - \mu)}{\sqrt{\frac{\alpha^2\sigma^2}{m} + \frac{\sigma^2}{m}}\sqrt{2}}\right)\right] = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{\frac{-(\alpha\mu - \mu)}{\sigma\sqrt{\frac{\alpha^2\sigma^2}{m} + \frac{\sigma^2}{m}}}}{\sqrt{2}}\right)\right]$$
$$= \phi\left(\frac{-(\alpha\mu - \mu)}{\sqrt{\frac{\alpha^2\sigma^2}{m} + \frac{\sigma^2}{m}}}\right) = \phi\left(\frac{-\mu(\alpha - 1)}{\sigma\sqrt{\frac{\alpha^2 + 1}{m}}}\right)$$

در ادامه با فرض $\sigma = \mu$ داریم:

$$\Pr[\alpha * CPUTime(S) < CPUTime(T)] = \phi\left(\frac{1 - \alpha}{\sqrt{\frac{\alpha^2 + 1}{m}}}\right)$$

ه) با جایگذاری $\alpha = 1.1$ و $m = 100$ داریم:

$$\phi(-0.6726727939963131) = 0.2506$$

و با جایگذاری $\alpha = 1.1$ و $m = 10000$ داریم:

$$\phi(-6.726727939963131) = 0$$

این اعداد بیانگر این است که به چه احتمالی پردازه اول بیشتر از پردازه دوم پردازنده را در اختیار میگیرد. پس لازم است این اعداد را در دو ضرب کنیم. نتیجه کلی این است که در m های کوچک، به احتمال بالایی یکی از پردازنده های سهم بیشتری از پردازنده میگیرد ولی با افزایش m به اندازه کافی، این احتمال به صفر میرسد.

و) در قسمت شبیه سازی هم اعداد ۰/۵ و ۰ به ترتیب برای $m = 100$ و $m = 10000$ بدست آمد که تایید کننده اعداد قسمت ه است.