به نام خدا

گزارش پروژه درس ارزیابی کارایی سیستم‌های کامپیوتری

۱۳۰۷
دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشجو: مهدی نیک نژاد

استاد: دکتر مرادیان

مرداد ۱۴۰۳

فهرست مطالب

مقدمه

می‌خواهیم در این پروژه، صف‌های M/M/1 و M/G/1 را با زبان پایتون پیاده‌سازی کنیم و سپس با مقادیر تحلیلی (محاسباتی) مقایسه کنیم. بدین صورت که نمودار مقدار متوسط زمان پاسخ مشتری را بر حسب نرخ ورود که از صفر تا نرخ سرویس تغییر می‌کند رسم می‌کنیم و نمودارهای مرتبط با $\mu = 1,2,8$ را در یک شکل رسم کرده و مقایسه می‌کنیم. همچنین مقدار حاصل از شبیه‌سازی را با مقدار تحلیل، مقایسه می‌کنیم. جزئیات این دو صف بدین شرح است:

صف M/M/1 :

- ورودی
  - نرخ ورود λ
  - نرخ سرویس μ
  - مدت زمان اجرای برنامه (به اندازه کافی زیاد؛ در این پروژه این عدد ۱۰۰۰ در نظر گرفته شده است)
- خروجی
  - متوسط زمان پاسخ مشتری

صف M/G/1 :

- ورودی
  - نرخ ورود λ
  - نرخ سرویس μ (توزیع زمان سرویس ارلانگ از مرتبه ۲ و پارامتر $\lambda/2$)
  - مدت زمان اجرای برنامه (به اندازه کافی زیاد؛ در این پروژه این عدد ۱۰۰۰ در نظر گرفته شده است)
- خروجی
  - متوسط زمان پاسخ مشتری

توضیح کد

ابتدا کتابخانه‌های لازم را وارد می‌کنیم؛ مانند : numpy, matplotlib و همچنین math و random.

بخش اول

در این بخش به پیاده‌سازی صف M/M/1 می‌پردازیم. در ابتدا اجزای این بخش را بیان می‌کنیم و بعد خروجی را تحلیل می‌کنیم.

- **کلاس MM1Queue**

کارکرد این کلاس شبیه‌سازی صف M/M/1 است و پارامترهای سازنده این کلاس نرخ ورود، نرخ سرویس و مدت زمان اجرای برنامه است. علاوه بر این، تعداد اتریبیوت هم دارد: صف از جنس لیست برای نگهداری نرخ ورود مشتریان حاضر در صف، جمع زمان‌های پاسخ، مجموع تعداد مشتریان و زمان جاری.

  - **تابع تولید زمان‌های بین دو ورود generate_interarrival_time**

  توزیع زمان بین دو ورود متوالی در صف M/M/1 از نوع نمایی است و این تابع این زمان را تولید می‌کند.

  - **تابع تولید زمان سرویس generate_service_time**

  این تابع هم به طور مشابه تابع قبلی از توزیع نمایی استفاده می‌کند.

  - **تابع simulate**

  این تابع بخش اصلی می‌باشد و به انجام شبیه سازی صف می‌پردازد و خروجی آن زمان پاسخ مشتری می‌باشد.
  ابتدا با تابع **generate_interarrival_time**، زمان ورود بعدی مشخص می‌شود و همچنین زمان مرگ بعدی را بی‌نهایت می‌گذاریم. سپس تا زمانی که به پایان مدت زمان اجرای برنامه نرسیده باشیم، ادامه می‌دهیم.

    - اگر زمان ورود بعدی قبل از زمان مرگ بعدی رخ دهد، مشتری را به صف اضافه می‌کنیم (زمان ورود آن را اضافه می‌کنیم) و بعد زمان ورود بعدی را تولید می‌کنیم. و بعد اگر فقط یک مشتری در صف باقیمانده باشد، زمان مرگ بعدی را تولید می‌کنیم.
    - و حالا اگر زمان مرگ بعدی قبل از زمان ورود بعدی باشد، ابتدا اولین مشتری را از صف برمی‌داریم و بعد زمان پاسخ به آن را محاسبه می‌کنیم و تعداد مشتریان سرویس گرفته را هم یکی زیاد می‌کنیم. و بعد اگر مشتری در حال انتظاری همچنان داشتیم، زمان مرگ بعدی را تولید می‌کنیم ولی اگر مشتری باقی‌نمانده باشد، زمان مرگ بعدی را بی نهایت می‌گذاریم. و اگر هم هیچ مشتری نمانده باشد که سرویس نگرفته باشد، باز هم زمان مرگ بعدی را بی نهایت می‌گذاریم.

  در انتها میانگین زمان پاسخ مشتری را با تقسیم جمع زمان های پاسخ بر تعداد مشتریان سرویس گرفته بدست می‌آوریم.

- **تابع analytical_response_time**

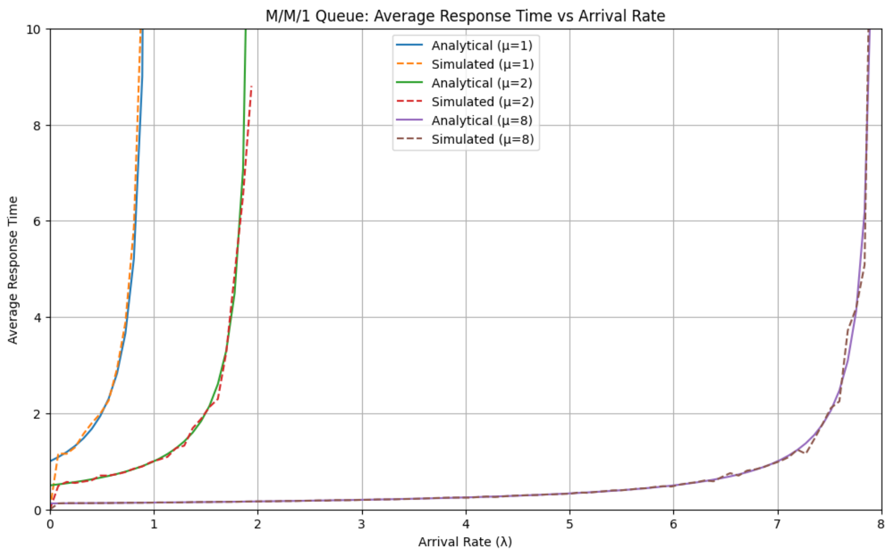
  این تابع زمان پاسخ در صف M/M/1 را بر اساس فرمولهایی که در اسلایدها داشتیم بدست می‌آورد. که برای صف M/M/1 برابر بود با :

$$E(T) = \frac{1}{\mu - \lambda}$$

- **تابع plot_response_time**

این تابع جهت نمایش نمودارها است. ابتدا با استفاده از np.linspace بازه‌ای از نرخ های ورودی می‌سازیم برای اینکه محور افقی باید از صفر تا μ باشد. با استفاده از تابع قبلی زمانهای پاسخ را طبق analytical بدست می‌آوریم. و بعد به ازای هر μ ، (ما کلا ۳ تا μ در نظر گرفتیم : ۱و۲و۸) ۲ حالت شبیه‌سازی و تحلیلی را رسم می‌کنیم.

در شکل زیر را می‌بینیم.



شکل 1خروجی بخش اول

همانطور که از شکل مشخص است، نتایج شبیه‌سازی با محاسبات تحلیلی تا حد خوبی مطابقت دارد. و همچنین می‌دانیم با نزدیک شدن λ به μ، میانگین زمان پاسخ به سرعت افزایش می یابد. وقتی λ با μ برابر می‌شود، صف بی نهایت داریم و یعنی تا قبل از آن سیستم پایدار است و زمان پاسخ محدود و معقول است اما بالاتر از آن سیستم ناپایدار می‌شود و زمان پاسخ بی نهایت می‌شود.

بخش دوم

در این بخش به پیاده‌سازی صف M/G/1 می‌پردازیم. در ابتدا اجزای این بخش را بیان می‌کنیم و بعد خروجی را تحلیل می‌کنیم.

- **کلاس MG1Queue**

این کلاس کاملا مشابه MM1Queue می‌باشد.

  - **تابع generate_interarrival_time**

  این تابع کاملا مشابه همنامش در بخش قبلی می‌باشد.

  - **تابع generate_service_time**

  این تایع زمان سرویس را برای مشتری ای تولید می‌کند که دارد از توزیع ارلانگ مرتبه ۲ با پارامتر μ/2 تبعیت می‌کند. زمان کل سرویس برابر با جمع دو تا توزیع نمایی است.

  - **تابع simulate**

  این تابع هم کاملا مشابه تابع simulate قبلی برای صف M/G/1 می‌باشد.

- **تابع analytical_mg1_response_time**

این تابع زمان پاسخ مشتری را بر حسب فرمولهایی که در اسلایدها داشتیم بیان می‌کند که طبق رابطه Pollaczek Khinchin داریم :

$$E(T) = E(W) + E(S) = \frac{\lambda \, \mathrm{E}[\mathrm{S}^2]}{2(1-p)} + \frac{1}{\mu}$$

$$E[S^2] = \frac{1}{\mu^2 + Var(S)}$$

تنها نکته‌ای که باید رعایت کنیم این است که اینجا μ برابر با μ/2 است. و همچنین Var(S) برای توزیع ارلانگ مرتبه ۲ با پارامتر μ/2 برابر با $\frac{2}{\left(\frac{\mu}{2}\right)^2}$ است. [۱]

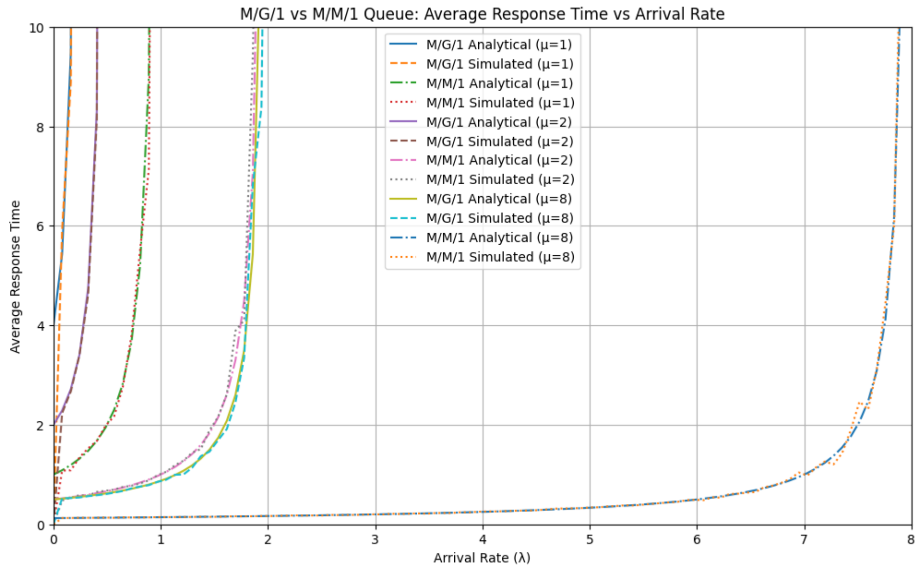
- **تابع analytical_mm1_response_time**

این تابع هم مشابه همانی است که در بخش اول برای صف *M/M/1* داشتیم.

- **تابع plot_response_times**

این تابع هم مشابه نسخه قبلی است و برای رسم نتایج *M/G/1* و همچنین *M/M/1* برای هر ۳ تا μ و در حالتهای شبیه سازی و تحلیلی می‌باشد. پس در مجموع ۱۲ نمودار ترسیم خواهد شد. که نتایج را در شکل زیر می‌بینیم.

[1] https://en.wikipedia.org/wiki/Erlang_distribution



شکل 2خروجی بخش دوم

همانطور که از شکل مشخص است، صف *M/G/1* با زمان سرویس *Erlang-2* معمولاً متوسط زمان پاسخ بالاتری نسبت به صف *M/M/1* دارد. هر دو صف با نزدیک شدن λ به μ ناپایدار می‌شوند، اما *M/G/1* زودتر ناپایدار می‌شود. به طور دقیق‌تر در $\lambda = \frac{1}{4}\mu$. (جزئیات بیشتر در پیوست قرار دارد) و همچنین نرخ سرویس بالاتر منجر به متوسط زمان پاسخ کمتر برای هر دو نوع صف می‌شود.

جمع بندی

در این پروژه ما هر دو صف M/M/1 و M/G/1 را پیاده سازی کردیم تا عملکرد آنها را در شرایط مختلف تحلیل کنیم. نکات زیر یافته‌های کلیدی در رابطه با متوسط زمان پاسخ را بیان می‌کند:

- در شبیه‌سازی‌های انجام‌شده، M/M/1 معمولاً **متوسط زمان‌های پاسخ** کمتری را در مقایسه با M/G/1 نشان می‌دهد
  برعکس، صف M/G/1 افزایش متوسط زمان پاسخ را با نزدیک شدن شدت ترافیک $\rho = \lambda/\mu$ به 1 نشان می‌دهد.
- در M/M/1 با افزایش **شدت ترافیک**، سیستم می تواند بار بالاتری را بدون افزایش قابل توجهی در متوسط زمان پاسخ تا زمانی که به ناپایداری نزدیک شود تحمل کند.
  ولی در M/G/1 با افزایش شدت ترافیک، متوسط زمان پاسخ می‌تواند به شدت افزایش یابد و منجر به زمان انتظار طولانی‌تر برای مشتریان شود.
- M/M/1 عموماً از نظر زمان پاسخگویی **کارایی** بیشتری دارد، به خصوص در شرایط بار سنگین.
  اما M/G/1، کارایی کمتری دارد، که می‌تواند منجر به صف‌های طولانی‌تر و زمان انتظار با شلوغ شدن سیستم شود.
  بنابراین، هنگام در نظر گرفتن کارایی و عملکرد، M/M/1 اغلب در بسیاری از کاربردهای عملی انتخاب ارجح است.

پیوست

$$E[S] = 2 \times \frac{1}{\frac{\mu}{2}} = \frac{4}{\mu}$$

$$Var(S) = \frac{2}{\left(\frac{\mu}{2}\right)^2} = \frac{8}{\mu^2}$$

$$E[S^2] = Var(S) + E[S]^2 = \frac{8}{\mu^2} + \frac{16}{\mu^2} = \frac{24}{\mu^2}$$

$$E[T] = E[S] + \frac{\lambda E[S^2]}{2(1-\rho)} = \frac{4}{\mu} + \frac{\lambda}{\cancel{2}} \cdot \frac{\overset{12}{\cancel{24}}}{\mu^2} \cdot \frac{1}{1 - \frac{4\lambda}{\mu}} = \frac{4}{\mu} + \frac{12\lambda}{\mu^2\left(1 - \frac{4\lambda}{\mu}\right)} = \frac{4\mu\left(1 - \frac{4\lambda}{\mu}\right) + 12\lambda}{\mu^2\left(1 - \frac{4\lambda}{\mu}\right)} =$$

$$\rho = \frac{\lambda}{\frac{\mu}{4}} = \frac{4\lambda}{\mu}$$

$$= \frac{4\mu \overset{-4\lambda}{\cancel{- 16\lambda + 12\lambda}}}{\mu^2\left(1 - \frac{4\lambda}{\mu}\right)} = \frac{4(\mu - \lambda)}{\mu^2\left(1 - \frac{4\lambda}{\mu}\right)} \rightarrow 1 - \frac{4\lambda}{\mu} = 0 \rightarrow$$

مجانب قائم در M/G/1 به صورت
تئوری و (Analytical) در
$\boxed{\lambda = \frac{1}{4}\mu}$ می‌باشد.

در حقیقت ۲ سرویس نمایی پشت سر همدیگر داریم که نرخ سرویس هر کدام $\frac{\mu}{2}$ هستند یعنی به طور میانگین زمان سرویس هر کدام از این نمایی‌ها برابر با $\frac{2}{\mu}$ است. پس میانگین زمان کل سرویس (۲ تا سرویس نمایی پشت سر همدیگر) برابر است با $2 * \frac{2}{\mu} = \frac{4}{\mu}$. و نرخ سرویس کل سیستم برابر با $\frac{\mu}{4}$ است.