دانشکده مهندسی برق و کامپیوتر دانشگاه تهران

گزارش فصل ۱: SINGLE SERVER QUEUING MODEL

امیرحسین مرادیان

۸۱۰۱۰۰۴۶۷

استاد درس:

دکتر خونساری

۳۰ خرداد ۱۴۰۱

مقدمه

سیستم به این صورت عمل می کند که مشتری وارد می شود و سرور را بیکار پیدا می کند و بلافاصله از سیستم سرویس می گیرد. لازم به ذکر است که زمان های سرویس دهی از مشتریان متوالی نیز متغیر های تصادفیIID هستند که مستقل از زمان های بین ورود می باشد. حال اگر یک مشتری وارد شود و سرور را مشغول بیابد، به انتهای صف می رود. زمانی که سرویس دهی به یک مشتری تکمیل می شود، سرور صف را چک کرده و در صورتی که مشتری در صف وجود داشته باشد، آن را به روش $FIFO$ انتخاب می کند.

شبیه سازی در حالت خالی صف و بیکار سرور آغاز می شود. در زمان صفر، ما شروع به انتظار برای ورود اولین مشتری خواهیم کرد که پس از اولین زمان بین ورود یا همان A_1، به جای زمان ۰ (که یک فرض مدل سازی احتمالاً معتبر، اما متفاوت است) رخ می دهد. ما میخواهیم این سیستم را تا زمانی شبیه سازی کنیم که تعداد مشخصی مشتری (که آن راnمی نامیم) تاخیر های خود را در صف تکمیل کنند. به عبارت دیگر، زمانی کهn مین مشتری وارد سرویس شود، شبیه سازی متوقف می شود ، زمان پایان شبیه سازی نیز یک متغیر تصادفی است.

برای سنجش عملکرد این سیستم سه کمیت را برآورد خواهیم کرد. اولین کمیت، میانگین تاخیر مورد انتظار در صف برایnمشتری خواهد بود .در این راستا، در یک اجرای معین از شبیه سازی، میانگین تاخیر واقعی مشاهده شده ازnمشتری به مشاهدات متغیر تصادفی بین ورود و زمان سرویس دهی بستگی دارد. این در حالی است که در اجرای دیگری از شبیه سازی، احتمالاً زمان های مختلفی برای زمان ورود سرویس دهی را شاهد خواهیم بود که این دلیلی بر درستی در نظر گرفتن میانگینnبه عنوان یک متغیر تصادفی است. چیزی که ما در این شبیه سازی به دنبال آن هستیم، برآورد میانگین تاخیر مورد انتظار مشتری است که آن را با $d(n)$ نمایش می دهیم و به صورت زیر آن را تعریف خواهیم کرد :

$$\hat{d}(n) = \frac{\Sigma_{i=1}^{n} D_i}{n} \qquad (۱)$$

یکی دیگر از این معیارها برای ارزیابی مدل میانگین مورد انتظار تعداد مشتریان در صف است (مشتریانی که به آنها سرویس دهی نمی شود)، که این کمیت را با $q(n)$ نمایش می دهیم. این یک نوع متوسط متفاوت از میانگین تاخیر در صف است، چرا که به جای مشتریان (گسسته بودن) زمان (پیوسته) گرفته می شود. بنابراین ما باید تعریف کنیم که منظور از این تعداد میانگین زمانی مشتریان در صف چیست. برای انجام این کار $Q(t)$ را تعداد مشتریان در صف در زمانt تعریف کرده و $T(n)$ را زمان مورد نیاز برای مشاهدهn تاخیر در صف در نظر میگیریم. به این صورت، برای هر زمان t بین ۰ و $Q(n)،T(t)$ یک عدد صحیح غیر منفی است. همچنین اگر p_i را نسبت مورد انتظار که بین ۰ و ۱ خواهد بود، از زمانی که $Q(t)$ برابر باiاست بگذاریم، به رابطه زیر خواهیم رسید :

$$\hat{q}(n) = \frac{\int_0^{T(n)} Q(t)dt}{T(n)} \qquad (۲)$$

سومین معیار ارزیابی عملکرد برای این سیستم، میزان شلوغی سرور است. استفاده مورد انتظار از سرور، نسبت زمان مورد انتظار در طول شبیه سازی (از زمان ۰ تا زمان (□)□) است که سرور مشغول است ، و بنابراین عددی بین ۰ و ۱ است و آن را با $u(n)$ نشان می دهیم. پس از یک شبیه‌سازی، تخمین ما از $u(n)$ برابر است با نسبت مشاهده‌شده در طول شبیه‌سازی که سرور مشغول است. اکنون $\hat{u}(n)$ را می‌توان مستقیماً از شبیه‌سازی با ذکر زمان‌هایی که در آن سرور وضعیت تغییر می‌کند (تغییر از بی‌کار به مشغول یا برعکس) و سپس انجام تفریق و تقسیم مناسب محاسبه کرد. با این حال، با تعریف "عملکرد مشغول" ساده تر است که به این کمیت به عنوان یک

میانگین زمان پیوسته، مشابه طول متوسط صف نگاه کنیم.

$$B(t) = \begin{cases} 1 & \text{if the server is busy at time t} \\ 0 & \text{if the server is idle at time t} \end{cases}$$

و بنابراین $\hat{u}(n)$ را می توان به عنوان نسبت زمانی بیان کرد که $B(t)$ برابر با ۱ است. با این حال با توجه به اینکه ارتفاع تابع $B(t)$ همواره بین ۰ و ۱ است و معیار مورد نظر ما را می توان به عنوان مساحت زیر تابع $B(t)$ در طول شبیه سازی در نظر گرفت، خواهیم داشت:

$$\hat{u}(n) = \frac{\int_0^{T(n)} B(t)dt}{T(n)} \quad (۳)$$

MATLAB Code

کد های متلب زده شده مرتبط با پروژه را در زیر میتوانید مشاهده کنید: در ابتدا میتوانید تایع arrive را که نشان دهنده رسیدن به سرور هست را مشاهده کنید:

arrive.m

```
function [timeNextEvent, numInQ, timeArrival, totalOfDelays, numCustsDelayed, serverStatus] = arrive(timeNextEvent, numInQ, timeArrival,...
    simTime, totalOfDelays, numCustsDelayed, serverStatus, meanInterarrival, outfile, meanService)
    global Q_LIMIT
    global BUSY
    global IDLE
%     Schedule next arrival.
    timeNextEvent(1) = simTime + expon(meanInterarrival);
%     Check to see whether server is busy.
    if(serverStatus == BUSY)
%     Server is busy, so increament number of customer in queue.
        numInQ = numInQ + 1;
%         Check to see whether an overflow condition exists.
        if(numInQ > Q_LIMIT)
%             The queue has overflowd, so stop the simulation.
            fprintf(outfile, '\nOverflow of the array timeArrival at');
            fprintf(outfile, ' time %f', simTime);
            exit(2);
        end
        %There is still room in the queue, so store the time of arrival of
        %the arriving customer at the (new) end of timeArrival.
        timeArrival(numInQ) = simTime;
    else
        %Server is idle, so arriving customer has a delay of zero. (The following two statements are
        % for program clarity and do not affect the results of the simulation.
        delay = 0;
        totalOfDelays = totalOfDelays + delay;
        % Increment the number of customers delayed, and make server busy.
        numCustsDelayed = numCustsDelayed + 1;
        serverStatus = BUSY;
%         Schedule a departure (service completion).
        timeNextEvent(2) = simTime + expon(meanService);
```

در شکل زیر ادامه کد را خواهید دید:

arrive.m

```
%     Schedule next arrival.
    timeNextEvent(1) = simTime + expon(meanInterarrival);
%     Check to see whether server is busy.
    if(serverStatus == BUSY)
%     Server is busy, so increament number of customer in queue.
        numInQ = numInQ + 1;
%         Check to see whether an overflow condition exists.
        if(numInQ > Q_LIMIT)
%             The queue has overflowd, so stop the simulation.
            fprintf(outfile, '\nOverflow of the array timeArrival at');
            fprintf(outfile, ' time %f', simTime);
            exit(2);
        end
        %There is still room in the queue, so store the time of arrival of
        %the arriving customer at the (new) end of timeArrival.
        timeArrival(numInQ) = simTime;
    else
        %Server is idle, so arriving customer has a delay of zero. (The following two statements are
        % for program clarity and do not affect the results of the simulation.
        delay = 0;
        totalOfDelays = totalOfDelays + delay;
        % Increment the number of customers delayed, and make server busy.
        numCustsDelayed = numCustsDelayed + 1;
        serverStatus = BUSY;
%         Schedule a departure (service completion).
        timeNextEvent(2) = simTime + expon(meanService);
        timeArrival = timeArrival;
    end
end
```

در کد زیر تابع جداشدن را از سرور میتوانید مشاهده کنید:

depart.m

```
function [numInQ, serverStatus,  timeArrival, totalOfDelays, numCustsDelayed, timeNextEvent] = ...
    depart(numInQ, serverStatus, simTime, timeArrival, totalOfDelays, numCustsDelayed, timeNextEvent, meanService)
    global Q_LIMIT
    global BUSY
    global IDLE
%     Check to see whether the queue is empty.
    if(numInQ == 0)
        % The queue is empty so make the server idle and eliminate the
        % departure (service completion) event from consideration.
        serverStatus = IDLE;
        timeNextEvent(2) = 1e+30;
    else
%         The queue is nonempty, so decrement the number of customers in queue.
        numInQ = numInQ - 1;
        % Compute the delay of the customer who is beginning service and
        % update the total delay accumulator.
        delay  = simTime - timeArrival(1);
        totalOfDelays = totalOfDelays + delay;
        % Increment the number of customers delayed, and schedule
        % departure.
        numCustsDelayed = numCustsDelayed + 1;
        timeNextEvent(2) = simTime + expon(meanService);
        % Move each customer in queue (if any) up one place.
        for i = 1: numInQ
            timeArrival(i) = timeArrival(i + 1);
        end
    end
end
```

در زیر میتوانید تابع exponential را مشاهده کنید

depart.m expon.m

```
function result = expon(mean)
    a = lcg(2)
    b = log(a)
    temp = -mean * b;
    result = temp;
end
```

در این بخش Initialize را مشاهده میکنید:

init.m | initialize.m

```
function [simTime, serverStatus, numInQ, timeLastEvent, ...
    numCustsDelayed, totalOfDelays, areaNumInQ, areaServerStatus, ...
    timeNextEvent] = initialize(meanInterarrival)
    global Q_LIMIT
    global BUSY
    global IDLE
    %Initialize the simulation clock.
    simTime = 0.0;
    %Initialize the state variables.
    serverStatus = IDLE;
    numInQ = 0;
    timeLastEvent = 0.0;
    %Initialize the statistical counters.
    numCustsDelayed = 0;
    totalOfDelays = 0;
    areaNumInQ = 0;
    areaServerStatus = 0;
    %Initialize event list. Since no customers are present, the departure
    %(service comletion) event is eliminated from consideration.
    timeNextEvent(1) = simTime + expon(meanInterarrival);
    timeNextEvent(2) = 1e+30;
end
```

Command Window

اکنون کد مربوط به بخش lcg را میتوانید مشاهده کنید:

lcg.m

```
function rnd = lcg(stream)
   load ('init.mat');
   zi = zrng(stream);
   lowprd = bitand(zi,65535, 'int32') * MULT1;
   hi31   = bitshift(zi,-16) * MULT1 + bitshift(lowprd,-16);
   zi     = (bitand(lowprd,65535, 'int32') - MODLUS) +  bitshift(bitand(hi31, 32767, 'int32'),16) ...
          + bitshift(hi31,-15);
   if  zi < 0
      zi  = zi + MODLUS;
   end
   lowprd = bitand (zi, 65535, 'int32') * MULT2;
   hi31   = bitshift(zi, -16) * MULT2 + bitshift(lowprd, -16);
   zi     = (bitand(lowprd, 65535, 'int32') - MODLUS) + bitshift(bitand(hi31, 32767, 'int32'), 16) ...
       + bitshift(hi31, -15);
   if zi < 0
      zi = zi + MODLUS;
   end
   zrng(stream) = zi;
   save('init.mat', 'MODLUS', 'MULT1', 'MULT2', 'zrng');
   rnd = bitor(bitshift(zi, -7), 1, 'int32') / 16777216.0;
end
```

در ادامه میتوانید بخش Report را مشاهده کنید:

report.m

```
function report(outfile, totalOfDelays, numCustsDelayed, areaNumInQ, simTime, areaServerStatus)
    %report generator function.
    %Compute and write estimates of desired meansures of performace.
    fprintf(outfile, '\n\nAverage delay in queue %11.3f minutes\n\n', ...
        totalOfDelays / numCustsDelayed);
    totalOfDelays
    numCustsDelayed
    fprintf(outfile, 'Average number in queue %10.3f\n\n', ...
        areaNumInQ / simTime);
    areaNumInQ
    simTime
    fprintf(outfile, 'Server utilization %15.3f\n\n', ...
        areaServerStatus / simTime);
    areaServerStatus
    simTime
    fprintf(outfile, 'Time simulation ended %12.3f minutes', simTime);


end


```

و همچنین تابع Timing را میبینید:

timing.m

```
function [simTime, nextEventType] = timing(simTime, outfile, numEvents, timeNextEvent)
    minTimeNextEvent = 1e+29;
    nextEventType = 0;
    for i = 1: numEvents
        if (timeNextEvent(i) < minTimeNextEvent)
            minTimeNextEvent = timeNextEvent(i);
            nextEventType = i;
        end
    end
%     check to see whether the event list is empty.
    if(nextEventType == 0)
%         The event list is empty, so stop simulation.
        fprintf(outfile, '\nEvent list empty at time %f', simTime);
        exit(1);
    end
%     The event list is not empty, so advance the simulation clock.
    simTime = minTimeNextEvent;
end


```

و در نهایت آخرین تابع تحت عنوان UpdateTimeArrival که زمان رسیدن به سرور را به روز رسانی میکند در زیر آمده است:

timing.m | updateTimeAvgStats.m

```
function  [timeLastEvent, areaNumInQ, areaServerStatus] = updateTimeAvgStats(timeLastEvent, areaNumInQ,...
    areaServerStatus, simTime, numInQ, serverStatus)
    global Q_LIMIT
    global BUSY
    global IDLE

    %Update area accumulators for time-average statistics.
    %Compute time since last event, and update last-event-time maker.
    timeSinceLastEvent = simTime - timeLastEvent;
    timeLastEvent      = simTime;
    %update area under number-in-queue function.
    areaNumInQ = areaNumInQ + numInQ * timeSinceLastEvent;
    % Update area under server-busy indicator function.
    areaServerStatus = areaServerStatus + serverStatus * timeSinceLastEvent;
end
```

تمامی کد های متلب در بالا به همراه توضیحات لازم آورده شده اند. میتوانید مشاهده کنید.