Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра программного обеспечения информационных технологий

ОТЧЁТ

по лабораторной работе №5-6

по курсу «Системный анализ и машинное моделирование»

Вариант 26

|  |  |
| --- | --- |
| Проверил:  Мельник Н.И. | Выполнил:  студент гр. 951003  А.В. |

Минск 2022

26) В трехканальной СМО с отказами интенсивность обслуживания первого канала 2, второго и третьего. Если первый канал свободен, то поступившая заявка занимает именно его. Найти среднее время пребывания заявки в системе Wc и абсолютную пропускную способность. Все потоки простейшие, =0,2, =0,8.

t1 = {0, 1} – кол-во заявок в канале №1

t2 = {0, 1} – кол-во заявок в канале №2

t3 = {0, 1} – кол-во заявок в канале №3

Общий вид кодировки состояния системы: {t1, t2, t3}



По графу построим аналитическую модель и, решив ее, определим вероятности состояний.

λ\*P000 = µ\*(2\*P100 + P010 + P001)

(λ+2\*µ) \*P100 = λ\*P000 + µ\*P110 + µ\*P101

(λ+3\*µ) \*P110 = λ\*P100 + µ\*P111 + λ\*P010

(λ+µ) \*P010 = 2\*µ\*P110 + µ\*P011

(µ+λ)\*P001 = µ\*P011 + 2\*µ\*P101

4\*µ\*P111 = λ\*(P110 +P101 + P011)

(λ+2\*µ) \*P011 = 2\*µ\*P111

(λ +3\*µ) \*P101 = µ\*P111+ λ\*P001

P000 + P100 + P110 + P010+ P001+ P111+ P001+ P101 = 1

P000 = 0.2\*(2\*P100 + P010 + P001) / 0.8

P100 = (0.8\*P000 + 0.2\*P110 + 0.2\*P101) / (0.8 + 2\*0.2)

P110 = (0.8\*P100 + 0.2\*P111 + 0.8\*P010) / (0.8+3\*0.2)

P010 = 2\*0.2\*P110 + 0.2\*P011

P001 = 0.2\*P011 + 2\*0.2\*P101

P111 = 0.8\*(P110 + P101 + P011) / (4\*0.2)

P011 = 2\*0.2\*P111/ (0.8+2\*0.2)

P101 = (0.2\*P111+ 0.8\*P001) /(0.8 + 3\*0.2)

P000 + P100 + P110 + P010+ P001+ P111+ P011+ P101 = 1

P000 = 0,0804

P100 = 0,0919

P110 = 0,1489

P010 = 0,0825

P001 = 0,0553

P111 = 0,3448

P011 = 0,1149

P101 = 0,0808

­­

A = λ \* Q = λ \* (1 – Pотк) = 0,8\*(1 – 0,3448) = 0,5241

Lc =1\*(P100+ P010+ P001) + 2\*(P110+ P011+ P101) + 3\*P111

= (0,0919+0,0825+0,0553) + 2\*(0,1489+0,1149+0,0808) + 3\*0,3448

= 1,9533

Wс = 1,9533 / 0,5241= 3,7269

Имитационная модель

P000 = 0.0844217914584460545077023181

P100 = 0.0909314556301299388529629474

P110 = 0.1476211482300569827350054971

P010 = 0.0815191764280653725585727503

P001 = 0.0555096612835845397906356804

P111 = 0.3452461433605647529167210002

P011 = 0.1141812533998962410981973764

P101 = 0.08056937020925611754020243

A: 0.5197642134164051868409893157

W system: 3.7508646430146463089842713705

public class Channel

{

public int Id { get; set; }

private Request? \_request;

private Generator \_generator;

public Channel(Generator generator, int id)

{

\_generator = generator;

Id = id;

}

public bool IsBusy() => \_request is not null;

public void ProcessRequest(decimal passedTime)

{

if (\_request is null) return;

\_request.RemainedTime -= passedTime;

if (Math.Abs(\_request.RemainedTime) <= decimal.Zero)

{

\_request.IsCompleted = true;

\_request = null;

}

}

public void AddRequest(Request request)

{

\_request = request;

var processingTime = \_generator.GetNext();

\_request.RemainedTime = processingTime;

\_request.ProcessingTime = processingTime;

}

public decimal RemainedToProcessTime => \_request?.RemainedTime ?? decimal.MaxValue;

}

public class Generator

{

private const int count = 10000;

private float Nu { get; }

private static Random Random = new Random();

private List<double> numbers;

public Generator(float nu)

{

Nu = nu;

GenerateNumbers(7, 209715120, 3);

}

public decimal GetNext()

{

var number = numbers[Random.Next(count)];

var value = (decimal)(-1.0f / Nu) \* (decimal)Math.Log(number);

return value;

}

private void GenerateNumbers(int a, int m, int r)

{

numbers = new List<double>(count);

for (int i = 0; i < count; i++)

{

r = (r \* a) % m;

numbers.Add((double)r / m);

}

}

}

public class InputStream

{

public Generator \_generator;

public decimal RemainedTime { get; set; }

public InputStream(Generator generator)

{

\_generator = generator;

RemainedTime = generator.GetNext();

}

public Request? Process(decimal passedTime)

{

RemainedTime -= passedTime;

if (Math.Abs(RemainedTime) <= decimal.Zero)

{

RemainedTime = \_generator.GetNext();

return new Request();

}

return null;

}

}

public class Request

{

public decimal ProcessingTime { get; set; }

public bool IsCompleted { get; set; } = false;

public bool IsDeclined { get; set; } = false;

public decimal RemainedTime { get; set; }

}

public class System

{

private readonly InputStream \_inputStream;

private List<Channel> \_channels;

public System(InputStream inputStream, List<Channel> channels)

{

\_inputStream = inputStream;

this.\_channels = channels;

States = new Dictionary<string, int>()

{

["000"] = 0,

["100"] = 0,

["110"] = 0,

["010"] = 0,

["001"] = 0,

["111"] = 0,

["011"] = 0,

["101"] = 0,

};

Ways = new Dictionary<string, HashSet<string>>()

{

["000"] = new(),

["100"] = new(),

["110"] = new(),

["010"] = new(),

["001"] = new(),

["111"] = new(),

["011"] = new(),

["101"] = new(),

};

}

public List<Request> Requests { get; set; } = new List<Request>();

public Dictionary<string, int> States { get; set; }

public Dictionary<string, HashSet<string>> Ways { get; set; }

public decimal TimePassed { get; set; }

public void Work(int ticks)

{

for (int i = 0; i < ticks; i++)

{

var state = GetState();

States[state]++;

var passedTime = GetMinTime();

TimePassed += passedTime;

foreach (var channel in \_channels)

{

channel.ProcessRequest(passedTime);

}

var request = \_inputStream.Process(passedTime);

if (request is not null)

{

Requests.Add(request);

var result = SendToChannel(request);

if (!result) request.IsDeclined = true;

}

}

}

public bool SendToChannel(Request request)

{

foreach (var channel in \_channels)

{

if (!channel.IsBusy())

{

channel.AddRequest(request);

return true;

}

}

return false;

}

public string GetState()

{

var states = new List<string>(3);

var indexes = new List<int>() {1, 2, 3};

indexes.ForEach(channel => states.Add(GetChannelState(channel)));

var state = string.Join("", states);

return state;

string GetChannelState(int index) =>

\_channels.Find(channel => channel.Id == index)!.IsBusy()

? "1"

: "0";

}

private decimal GetMinTime()

{

var minInChannel = \_channels.Min(channel => channel.RemainedToProcessTime);

return minInChannel > \_inputStream.RemainedTime

? \_inputStream.RemainedTime

: minInChannel;

}

}

var inputStream = new InputStream(new Generator(lambda));

var firstChannel = new Channel(new Generator(2 \* nu), 1);

var secondChannel = new Channel(new Generator(nu), 2);

var thirdChannel = new Channel(new Generator(nu), 3);

var system = new Lab6.System(inputStream, new List<Channel> {firstChannel, secondChannel, thirdChannel});

var ticks = 1000000;

system.Work(ticks);

Console.WriteLine($"A = {system.Requests.Count(request => request.IsCompleted) / system.TimePassed}");

var completedRequests = system.Requests

.Where(request => request.IsCompleted && !request.IsDeclined);

Console.WriteLine($"W system: {completedRequests.Sum(request => request.ProcessingTime) / completedRequests.Count()}");