Федеральное государственное бюджетное образовательное учреждение
высшего образования «Сибирский государственный университет
телекоммуникаций и информатики»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

по дисциплине
«Теория функционирования распределённых вычислительных систем»

Выполнил:
Студент гр. ИВ-622
Тимофеев Д.А.

Проверила:
Преподаватель Кафедры ВС
Ткачева Т.А.

Новосибирск 2020

Оглавление

1.Цель роботы

Исследовать функцию оперативной надежности, восстановимости и коэффициента готовности распределенных вычислительных систем со структурной избыточностью.

Имеется распределенная вычислительная система (ВС) укомплектованная N одинаковыми элементарными машинами (ЭМ). Основная подсистема (вычислительное ядро) ВС состоит их n ЭМ, n – N элементарных машин составляют структурную избыточность. Заданы λ – интенсивность потока отказов любой из N элементарных машин ([λ] = 1/ч), m – количество восстанавливающих устройств восстанавливающей системы и μ – интенсивность потока восстановления элементарных машин одним восстанавливающим устройством ([μ] = 1/ч).

При анализе надежности ВС в стационарном режиме работы используются такие показатели как функция R* (t) оперативной надежности, функция U*(t) оперативной восстановимости и коэффициент S готовности.

В рамках лабораторной работы требуется выполнить нижеследующие задания.

1.Написать программу расчета функции R*(t) оперативной надежности, функции U* (t) оперативной восстановимости и коэффициента S готовности ВС со структурной избыточностью.

2. Построить график зависимости функции R*(t) оперативной надежности для следующих значений параметров: N = 10; n ∈ {8, 9, 10}; λ = 0,024 1/ч; μ = 0,71 1/ч; m = 1; t = 0, 2, 4, …, 24 ч.

3.Построить график зависимости функции U*(t) оперативной восстановимости для следующих значений параметров: N = 16; n ∈ {10, 11, …, 16}; λ = 0,024 1/ч; μ = 0,71 1/ч; m = 1; t = 0, 2, 4, …, 24 ч.

4.Заполнить таблицу значений показателя S для следующих значений параметров: N = 16; λ = 0,024 1/ч; μ = 0,71 1/ч.

2.Теория

2.1Определение основных параметров и основные формулы

Функция надежности R(t) – вероятность того, что производительность ВС, начавшей функционировать в состоянии i $(n \leq i \ll N)$ на промежутке времени 0, t, равна производительности основной подсистемы.

$$R(t) = P\{\forall\tau \in [0,t) \rightarrow \cap\,(\tau) = A_n n\omega | n \leq i \leq N\}$$

Функция восстанови мости $\boldsymbol{U(t)}$ – вероятность того, что в ВС, имеющей начальное состояние i $0 \leq i \ll n$, будет восстановлен на промежутке времени 0, t уровень производительности равный производительности основной подсистемы.

$$U(t) = 1 - P\{\forall\tau \in [0,t) \rightarrow \cap\,(\tau) = 0 | 0 \leq i < n\}$$

Функция готовности $\boldsymbol{S}$ (t) – вероятность того, что производительность системы, начавшей функционировать в состоянии i $0 \leq i \ll N$, равна в момент времени $t \geq 0$ производительности основной подсистемы.

$$S(t) = P\{\cap\,(t) = A_n n\omega | i \in E_0^N\}$$

Предельные значения показателей при $t \rightarrow \infty$ будут характеризовать надёжность ВС в стационарном режиме работы. Однако для данного режима такие показатели, как $R(t)$ и $U(t)$, не информативны $\lim t\rightarrow\infty\ R\ t = 0$, $\lim t\rightarrow\infty\ U\ t = 1$ Для оценки производительности ВС на промежутке времени при длительной эксплуатации используются функции $R*(t)$ и $U*(t)$ оперативной надёжности и восстановимости ВС.

В отличие от функций надёжности и восстановимости, функция готовности, введённая для переходного режима, может быть использована и в стационарном режиме работы ВС. В самом деле:

$$\lim_{t\to\infty} S(t) = \sum_{j=n}^{N} \lim_{t\to\infty} P_j(i,t) = \sum_{j=n}^{N} P_j = S$$

Причём предел S не зависит от начального состояния системы $i \in E0\ N$. Величину S называют коэффициентом готовности. Он является самым распространённым показателем для стационарного режима функционирования ВС.

2.2Модель функционирования ВС со структурной избыточностью

Практически приемлемым для вычисления показателей является подход, основанный на классическом аппарате массового обслуживания и методах приближенных вычислений. Схема подхода:

1. Составляются дифференциальные уравнения для вероятностей состояний системы с учётом подмножества поглощающих состояний.
2. Задаются начальные состояния.
3. Система дифференциальных уравнений с помощью преобразования Лапласа сводится к алгебраической.
4. Определяется решение алгебраической системы уравнений, причем решение выражается через полиномы, вычисляемые рекуррентно.
5. Доказываются свойства корней полиномов, позволяющее приближённо вычислять их значения.
6. После обращения преобразования Лапласа выписываются формулы для показателей качества функционирования ВС.
7. Для получения числовых значений показателей составляются программы.

К методике расчета предъявляют требования:

1. Приемлемость к большемасштабным ВС
2. Адекватность реальному процессу работы ВС или реализация принципов квазианалогии
3. Единообразие методов исследования функционировая ВС при произвольном количестве ЭМ
4. Простота численного анализа функционирования ВС при произвольном количестве ЭМ
5. Возможность выявления общих закономерностей, которые отражают достигнутый и перспективный уровни технологии ВТ.

3.Ход работы

Была написана программа для исследований функций оперативной надежности, восстановимости и коэффициента готовности распределенных вычислительных систем со структурной избыточностью. Так же для расчета

Ниже представлены графики зависимости функции оперативной надежности для определенных значений параметров, а так же график функции оперативной восстановимости

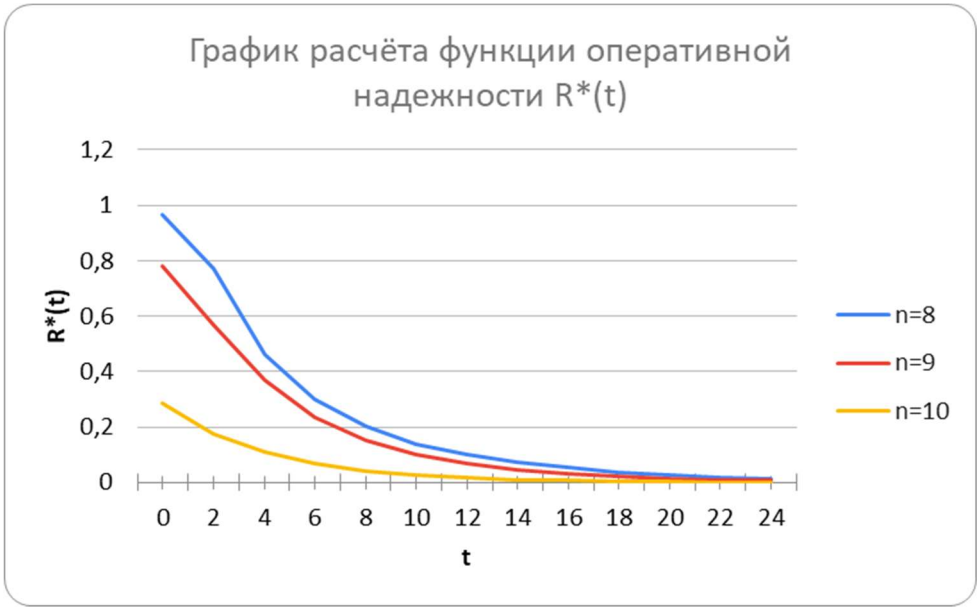

Рисунок 1График оперативной надежности R(t)

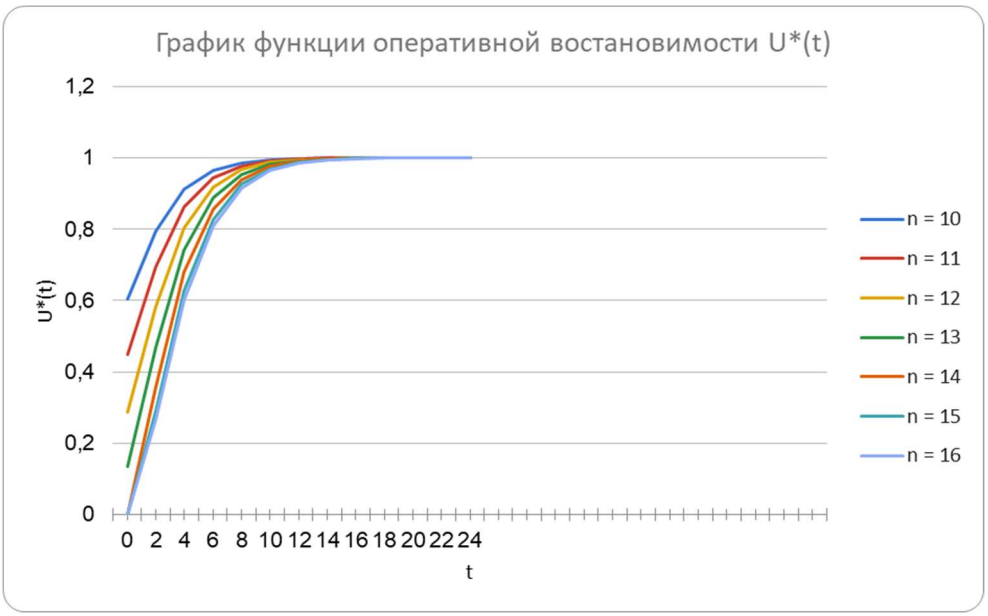

Рисунок 2График оперативной восстановимости U(t)

Ниже представлена таблица с данными, которые у нас получились после выполнения программы для оперативной надежности

n=8	n=9	n=10
0,967	7,79E-01	2,84E-01
0,77	5,68E-01	1,76E-01
0,464	3,70E-01	1,09E-01
0,299	2,34E-01	6,70E-02
0,203	1,50E-01	4,20E-02
0,138	0,099	2,60E-02
0,099	0,066	1,60E-02
0,073	0,044	0,01
0,053	0,03	0,006
0,038	0,02	0,004
0,027	0,014	0,002
0,019	0,009	0,001
0,013	0,006	0,001

А так же данные для функции оперативной восстановимсти.

n = 10	n = 11	n = 12	n = 13	n = 14	n = 15	n = 16
0,605	0,449	2,87E-01	1,34E-01	0,00E+00	0,00E+00	0
0,795	0,696	0,583	4,68E-01	3,58E-01	2,89E-01	0,268
0,912	0,863	0,805	7,42E-01	6,80E-01	6,29E-01	0,602
0,965	0,944	0,918	0,888	8,57E-01	8,28E-01	0,809
0,987	0,978	0,967	0,954	0,94	0,926	0,915
0,995	0,992	0,988	0,982	0,976	0,969	0,964
0,998	0,997	0,995	0,993	0,991	0,988	0,985
0,999	0,999	0,998	0,997	0,997	0,995	0,994
1	1	0,999	0,999	0,999	0,999	0,998
1	1	1	1	1	0,999	0,999
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

Заполнил таблицу значений S для следующих значений: N = 16; λ = 0,024 1/ч; μ = 0,71 1/ч

	1	16
10	0,9979	0,9999
11	0,9937	0,9999
12	0,9824	0,9999
13	0,9544	0,9999
14	0,8907	0,9999
15	0,7561	0,9989
16	0,2253	0,9673

4.Заключение

В результате лабораторной работы была написана программа, которая рассчитала функцию оперативной надежности и оперативной восстановимости , а так же коэффициента S готовности ВС со структурной избыточностью

По графику оперативной надежности можно сделать вывод, что при уменьшении потока входящих данных N время на выполнения будет увеличиваться, но во втором графике совсем все наоборот при увеличении потока данных оперативной восстановимости время будет стремительно расти.

5. Листинг

```
#include <iostream>

#include <fstream>

#include <cmath>

#include <iomanip>

#define DELTA(x) ( (x < 0) ? false : true)

#define MLEN 25

#define LMAX 10

#define RMAX 9

using namespace std;

/* Function for calculating factarial */

int fac(int x)

{

        int res = 1, tmp = 1;

        if (x == 0) {

                return 1;

        } else {

        while (x != 0) {

                res *= tmp;

                tmp++;

                x--;

        }

}

        return res;

}

/* Function for calculating chance l reparing by t time */

double U_l(int N, int m, int l, int i, double t, double nu)

{

        double result = 0;

        if (DELTA(m - N + i) == true) {

                result += pow(N - i, l) * pow(1 / M_E, (N - i) * nu * t);
```

```
        } else {
                result += pow(m, l) * pow(1 / M_E, m * nu * t);
        }
        result *= pow(nu * t, l) / fac(l);
        return result;
}
/* Function for calculating chance r fainlures by t time */
double Pi_r(int r, int i, double t, double lyambda)
{
        double result = 0;
        result = pow(1 / M_E, i * lyambda * t);
        return (result *= pow(i * lyambda * t, r) / fac(r));
}
/* Function for calculating chance l reparing and r failures by t time */
double Q(int N, int n, int m, int i, double t, double nu, double lyambda)
{
        double result = 0;
        for (int l = 0; l <= LMAX; l++) {
                double temp = 0;
                for (int r = 0; r <= i - n + l; r++) {
                        temp += Pi_r(r, i, t, lyambda);
        }
                result += temp * U_l(N, m, l, i, t, nu);
        }
        return result;
}
/* Function for calculating chance Pi */
double P(int N, int i, double nu, double lyambda)
{
        double result = 0, temp = 0;
        result = pow(nu / lyambda, i) / fac(i);
```

```
        for (int l = 0; l <= N; l++) {
                temp += pow(nu / lyambda, l) / fac(l);
        }
        result /= temp;
        return result;
}
/* Function for calculating operational safaty R */
double R(int N, int n, int m, double t, double nu, double lyambda)
{
        double result = 0;
        for (int i = n; i <= N; i++) {
                result += P(N, i, nu, lyambda) * Q(N, n, m, i, t, nu, lyambda);
}
        return result;
}
/* Function for calculating down rating safaty R */
double R_(int N, int n, int m, double t, double nu, double lyambda)
{
        double result = 0, temp = 0;
        for (int r, i = n; i <= N; i++) {
                temp = P(N, i, nu, lyambda);
                for (r = 0; r <= i - n; r++) {
                        temp *= Pi_r(r, i, t, lyambda);
}
                result += temp;
}
        return result;
}
/* Function for calculating operational recovery U */
double U(int N, int n, int m, double t, double nu, double lyambda)
{
```

```
        double result = 0;
        for (int i = 0; i < n; i++) {
                double temp = 0;
                for (int r = 0; r <= RMAX; r++) {
                        double temp2 = 0;
                        for (int l = 0; l < n - i + r; l++) {
                                temp2 += U_l(N, m, l, i, t, nu);
                        }
                        temp += temp2 * Pi_r(r, i, t, lyambda);
}
                result += temp * P(N, i, nu, lyambda);
}
        return (1 - result);
}
/* Function for calculating down rating operational recovery U */
double U_(int N, int n, int m, double t, double nu, double lyambda)
{
        double result = 0, temp = 0;
        for (int l, i = 0; i < n; i++) {
                temp = P(N, i, nu, lyambda);
                for (l = 0; l < n - i; l++) {
                        temp *= U(N, m, l, i, t, nu);
}
                result += temp;
}
        return (1 - result);
}
/* Functiona for calculating rate of availability */
double S(int N, int n, double nu, double lyambda)
{
        double result = pow(lyambda, N - n + 1) * pow(1 / (lyambda + nu), N - n + 1);
```

```
        return (1 - result);
}
int main()
{
        ofstream outfile;
        /* Task 1 */
        int N = 0, m = 0;
        double lyambda = 0.0, nu = 0.0, result = 0.0, t = 0.0;
        cout << " Task 1 " << endl;
        outfile.open("g_Safaty_R.txt");
        N = 10, m = 1, lyambda = 0.024, nu = 0.71;
        for (t = 0.0; t < 25.0; t += 2.0) {
                outfile << t;
                for (int n = 8; n < 11; n++) {
                        result = R(N, n, m, t, nu, lyambda);
                        outfile << setw(20) << result;
}
        outfile << endl;
}
        outfile.close();
        /* Task 2 */
        cout << " Task 2 " << endl;
        outfile.open("g_Repair_U.txt");
        N = 16;
        for (t = 0.0; t < 25.0; t += 2.0) {
                outfile << t;
                for (int n = 10; n < 17; n++) {
                        result = U(N, n, m, t, nu, lyambda);
                        outfile << setw(10) << " " << result;
}
        outfile << endl;
```

```
}

        outfile.close();

        /* Task 3 */

        cout << " Task 3" << endl;

        for (int n = 11; n < 17; n++) {

                outfile << n << setw(10) << U(N, n, 1, 0, nu, lyambda) << setw(15) <<S(N,
        n,nu,lyambda)<<endl;

}

        outfile.close();

        return 0;

}
```