Надёжность информационных систем. Лабораторная работа №2. Оценка и статистическое моделирование показателей надёжности восстанавливаемых информационных систем

Макаров П. А.

Содержание

1 Введение **1**

2 Краткая теория **2**

3 Практическое задание **2**
3.1 Системный анализ и актуализация SLA 3
3.2 Аналитический расчёт 3
3.3 Статистическое моделирование на Python 3

4 Требования к отчёту **5**

5 Контрольные вопросы **6**

Список источников **6**

1 Введение

В Лекции 3 конспекта [1] рассматриваются показатели надёжности *восстанавливаемых* объектов. В частности, в данном тексте присутствует утверждение: *“Для типичного современного сервера* $K_г = 0.99$*, что означает примерно 3.5 суток простоя в год”*, а также таблица классификации систем по уровню надёжности, где уровень 0.99999 (пять девяток) соответствует 5 минутам простоя. Сегодня стандарт 0.99 (две девятки) считается недопустимым для production-среды и применяется только для тестовых кластеров.

Кроме того, в современных реалиях, с развитием облачных вычислений, микросервисной архитектуры и систем оркестрации (например, Kubernetes или K8s, см. [2]), понятие «восстановления» перешло с уровня ручного ремонта аппаратуры на уровень *автоматического самовосстановления* (self-healing) системного и программного обеспечения. При этом соглашения об уровне обслуживания (SLA — Service Level Agreements) для базовых IaaS/PaaS сервисов (Infrastructure as a Service / Platform as a Service) начинаются от 99.9% и доходят до 99.999%. Более того, время восстановления ($t_в$ или MTTR — Mean Time To Recovery) для контейнеризированных приложений измеряется миллисекундами и секундами, а не часами.

Цель работы

1. Изучить количественные показатели надёжности восстанавливаемых ИС: параметр потока отказов $\omega(t)$, наработку на отказ $t_{\text{ср}}$, интенсивность восстановления μ и коэффициент готовности $K_{\text{г}}$.

2. Провести *критический анализ* устаревших справочных данных из Лекции 3 [1] и актуализировать их применительно к современным облачным и распределённым системам.

3. Исследовать влияние закона распределения времени безотказной работы (экспоненциальный и Вейбулла) на оценку долгосрочной надёжности.

4. Разработать программу на Python для статистического моделирования процессов отказов и восстановлений, визуализации состояния системы и расчёта эмпирического коэффициента готовности.

2 Краткая теория

Восстанавливаемыми согласно ГОСТ Р 27.102-2021 [3] называются объекты, восстановление работоспособного состояния которых в рассматриваемой ситуации предусмотрено технической и конструкторской документацией. Основные показатели таких объектов связывают время безотказной работы и время восстановления:

- **Средняя наработка на отказ** $t_{\text{ср}}$ (MTBF — Mean Time Between Failures) — математическое ожидание времени между соседними отказами.
- **Среднее время восстановления** $t_{\text{в}}$ (MTTR).
- **Интенсивность восстановления** $\mu = 1/t_{\text{в}}$.
- **Коэффициент готовности** $K_{\text{г}}$ — вероятность того, что объект находится в работоспособном состоянии в данный момент времени:

$$K_{\text{г}} = \frac{t_{\text{ср}}}{t_{\text{ср}} + t_{\text{в}}}. \tag{1}$$

В Лекции 3 [1] отмечено, что для электронных систем часто применяется **экспоненциальная модель** ВБР $P(t) = e^{-\lambda t}$, где $\lambda = \text{const}$. Однако, для систем, подверженных износу, часто применяется **модель Вейбулла**:

$$P(t) = e^{-\alpha t^{\beta}}, \tag{2}$$

где $\beta < 1$ — приработка (ранние отказы), $\beta = 1$ — режим нормальной эксплуатации, $\beta > 1$ — износ (примеры: старение SSD, деградация батарей, усталость металлов и т. п.).

3 Практическое задание

Задание включает в себя системный анализ, аналитический расчёт и компьютерное моделирование.

3.1 Системный анализ и актуализация SLA

Проанализируйте таблицу классификации систем по уровню надёжности из Лекции 3 [1]. Замените устаревшие данные на современные. Составьте соответствующую таблицу, сопоставив уровни коэффициентов готовности с примерами современных IaaS/PaaS сервисов и применяемых в них архитектурных решений и ПО.

Таблица 1: Пример таблицы для заполнения

Уровень $K_г$	**Примеры современных архитектур**
0.99	Виртуальная машина без кластеризации, локальный SQLite, ...
0.999	Kubernetes Deployment с 2 репликами, managed PostgreSQL, ...
0.9999	?
0.99999	?

Для каждого уровня укажите типичные значения MTBF и MTTR, обеспечивающие такую готовность.

3.2 Аналитический расчёт

Рассмотрим современный микросервисный шлюз (API Gateway), который работает в связке с кэшем Redis (REmote DIctionary Server) [4]. Известно, что:

- MTBF шлюза составляет 2 000 часов, MTTR (время автоматического перезапуска контейнера оркестратором) — 0.01 часа (36 секунд).
- MTBF кэша принимаем равным 5 000 часов, MTTR — 0.005 часов (18 секунд).

Рассчитайте следующие показатели.

1. Индивидуальные коэффициенты готовности $K_г^{(1)}$ и $K_г^{(2)}$.
2. Общий коэффициент готовности системы $K_г^{(sys)}$ при последовательном (с точки зрения логики работы сервиса) соединении.

Сравните полученный результат для $K_г^{(sys)}$ с требованием SLA уровня 99.99%. Достаточно ли текущих параметров? Если нет, как нужно изменить MTTR, чтобы достичь цели?

3.3 Статистическое моделирование на Python

Необходимо написать скрипт на Python, который моделирует эксплуатацию *одного восстанавливаемого узла* (например, физического сервера или критичного микросервиса) в течение $T_{max} = 8\,760$ часов (1 год).

Требования к скрипту

- Необходимо сгенерировать последовательность интервалов безотказной работы (TTF) и времени восстановления (TTR).

- Требуется реализовать возможность выбора закона распределения TTF: экспоненциальный (постоянная интенсивность отказов) и Вейбулла (с учётом старения, $\beta > 1$).
- Следует рассчитать эмпирический коэффициент готовности $K_{\text{г}}$ как отношение времени в состоянии «Up» к общему времени работы объекта.
- Построить временну́ю диаграмму состояний системы (State Transition Plot) и график накопления времени простоя (Downtime Accumulation Plot).

Пример кода

```
import math
import numpy as np
import matplotlib.pyplot as plt

# --- Исходные данные ---
T_max = 8760        # Время моделирования (1 год), часы
mtbf_init = 1000    # Средний параметр наработки (масштаб), часы
mttr = 0.5          # Среднее время восстановления, часы
beta_weibull = 1.5 # Параметр формы Вейбулла (>1 означает износ/старение)

def simulate_system(mtbf, mttr, T_max, distribution='exponential',
↪  beta=1.0):
    t_current = 0
    states = []     # Для графика: 1 - Up, 0 - Down
    times = []      # Метки времени

    while t_current < T_max:
        # 1. Генерация времени до отказа (TTF)
        if distribution == 'exponential':
            ttf = np.random.exponential(scale=mtbf)
        elif distribution == 'weibull':
            # Масштаб Вейбулла (eta) связан с MTBF через гамма-функцию
            eta = mtbf / math.gamma(1 + 1/beta)
            ttf = eta * np.random.weibull(a=beta)

        # 2. Генерация времени восстановления (TTR)
        ttr = np.random.exponential(scale=mttr)

        # 3. Запись состояния "Up"
        t_up_end = min(t_current + ttf, T_max)
        times.append(t_current); states.append(1)
        times.append(t_up_end); states.append(1)

        t_current = t_up_end
        if t_current >= T_max:
            break

        # 4. Запись состояния "Down" (Восстановление)
        t_down_end = min(t_current + ttr, T_max)
        times.append(t_current); states.append(0)
```

```
            times.append(t_down_end); states.append(0)

            t_current = t_down_end

    return np.array(times), np.array(states)

# Запуск моделирования
np.random.seed(42)
times_exp, states_exp = simulate_system(mtbf_init, mttr, T_max,
↪  'exponential')
times_wei, states_wei = simulate_system(mtbf_init, mttr, T_max, 'weibull',
↪  beta_weibull)

# Расчёт эмпирического коэффициента готовности
K_g_exp = np.sum(np.diff(times_exp) * states_exp[:-1]) / T_max
K_g_wei = np.sum(np.diff(times_wei) * states_wei[:-1]) / T_max

print(f"Эмпирический K_г (Экспоненциальный): {K_g_exp:.5f}")
print(f"Эмпирический K_г (Вейбулл, beta={beta_weibull}): {K_g_wei:.5f}")

# Визуализация состояния системы
plt.figure(figsize=(14, 6))
plt.step(times_exp, states_exp, where='post', label='Экспоненциальное
↪  ($\\lambda=const$)', alpha=0.8)
plt.step(times_wei, states_wei, where='post', label=f'Вейбулла
↪  ($\\beta={beta_weibull}$, износ)', alpha=0.8)
plt.yticks([0, 1], ['Down (Отказ/Восстановление)', 'Up (Работа)'])
plt.xlabel('Время, часы')
plt.title('Моделирование состояний восстанавливаемой системы во времени')
plt.legend()
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.savefig('system_states_simulation.png', dpi=300)
```

4 Требования к отчёту

Отчёт должен содержать:

1. Титульный лист, цель работы и краткие теоретические положения.
2. Таблицу с актуализированной классификацией систем по уровню готовности (SLA) с указанием современных архитектурных решений и типичных значений MTTR.
3. Результаты аналитических расчётов с выводами о соответствии SLA.
4. Листинг программного кода с комментариями.
5. Графики, сгенерированные скриптом.
6. Выводы о том, как современные средства оркестрации снижают $t_{\text{в}}$, компенсируя тем самым конечную надёжность ИС.

В частности, студенту необходимо объяснить, почему при $\beta > 1$ (распределение Вейбулла) количество отказов во второй половине года увеличивается, и как это влияет на необходимость превентивного обслуживания (Predictive Maintenance)?

5 Контрольные вопросы

1. В чём принципиальное отличие показателей надёжности для восстанавливаемых и невосстанавливаемых систем?
2. Дайте определение коэффициента готовности $K_{\text{г}}$.
3. Почему в современных облачных системах параметр $t_{\text{в}}$ (MTTR) стремится к нулю, и какие системные механизмы это обеспечивают?
4. Как изменяется интенсивность отказов $\lambda(t)$ в модели Вейбулла при $\beta > 1$ и $\beta < 1$? Какой физический смысл имеют эти параметры для серверного оборудования и ПО?
5. В Лекции 3 [1] утверждается, что для типичного сервера $K_{\text{г}} = 0.99$. Почему сейчас такой показатель считается критически низким для бизнес-приложений?
6. Как параметр потока отказов $\omega(t)$ связан с наработкой на отказ $t_{\text{ср}}$ для восстанавливаемой системы в установившемся режиме?

Список источников

1. *Расулова С. С.* Надёжность информационных систем. — Ташкент : Ташкентский университет информационных технологий, 2007. — С. 118.
2. Kubernetes Documentation / Cloud Native Computing Foundation. — URL: https://kubernetes.io/docs (дата обр. 14.09.2026).
3. ГОСТ Р 27.102-2021. Надёжность в технике. Надёжность объекта. Термины и определения. — Москва, 2021.
4. Redis Documentation / Redis Ltd. — URL: https://redis.io/docs (дата обр. 14.09.2026).

Актуальные отраслевые стандарты и документация

- AWS Well-Architected Framework (Reliability Pillar).
- Yandex Cloud SLA.
- Kubernetes Documentation: Liveness, Readiness, and Startup Probes.