Надёжность информационных систем. Лабораторная работа №4. Структурное резервирование: виды, схемы и расчёт надёжности

Макаров П. А.

Содержание

1 Введение **1**

2 Краткая теория **2**

3 Практическое задание **3**
3.1 Системный анализ и актуализация видов резервирования 3
3.2 Аналитический расчёт 3
3.3 Статистическое моделирование на Python 4

4 Требования к отчёту **9**

5 Контрольные вопросы **10**

Список источников **10**

1 Введение

В Лекции 6 конспекта [1] рассматривается один из наиболее эффективных методов повышения надёжности информационных систем — *структурное резервирование*, то есть введение в систему избыточных элементов, способных взять на себя выполнение рабочих функций при отказе основных. Автор классифицирует виды резервирования по схеме включения (постоянное, раздельное, замещением, скользящее), по режиму работы резервных элементов (нагруженный, облегчённый, ненагруженный), а также по кратности (целая и дробная, включая мажоритарное).

Цель работы

1. Изучить классификацию видов структурного резервирования и расчётные формулы для вероятности безотказной работы (ВБР) резервированных систем.

2. Провести *сопоставление* классических видов резервирования из Лекции 6 [1] с современными архитектурными решениями для распределённых систем.

3. Выполнить сравнительный аналитический расчёт ВБР системы при различных схемах резервирования (общее, раздельное, скользящее, мажоритарное).

4. Разработать программу на Python для моделирования отказов в резервированной системе и эмпирической проверки теоретических формул методом Монте-Карло.

2 Краткая теория

Определение и классификация

Резервированием называют метод повышения надёжности объекта путём введения избыточности — дополнительных элементов и устройств, способных взять на себя выполнение рабочих функций при отказе основных.

Классификация по схеме включения

- **Постоянное (общее) резервирование** — резервные элементы функционируют наравне с основными с самого начала работы системы.
- **Раздельное резервирование** — резервируются отдельные части (элементы) объекта.
- **Резервирование замещением** — резервный элемент подключается только после отказа основного; требует переключающих устройств.
- **Скользящее резервирование** — группа основных элементов резервируется одним или несколькими резервными, каждый из которых может заменить *любой* отказавший.

Классификация по режиму работы резерва

- **Нагруженный резерв** — резервный элемент работает в том же режиме, что и основной (отказывает с той же интенсивностью).
- **Облегчённый резерв** — резервный элемент работает в менее нагруженном режиме.
- **Ненагруженный резерв** — резервный элемент не несёт нагрузки и в период резервирования отказывать не должен (идеальная надёжность).

Основные расчётные формулы

Для экспоненциального закона распределения ($\lambda = \text{const}$).

1. Общее резервирование с постоянно включённым нагруженным резервом и целой кратностью m:

$$P_{\text{sys}}(t) = \left[1 - \left(1 - e^{-\lambda t}\right)^{m+1}\right]^{n}, \tag{1}$$

где n — число основных элементов, m — кратность резервирования.

2. Раздельное резервирование с нагруженным резервом и целой кратностью m_i для i-го элемента:

$$P_{\text{sys}}(t) = \prod_{i=1}^{n} \left[1 - \left(1 - e^{-\lambda_i t}\right)^{m_i+1}\right]. \tag{2}$$

3. Мажоритарное резервирование (частный случай дробной кратности, $m = 2$, $n = 1$, голосование «2 из 3»):

$$P_{\text{sys}}(t) = P_{\text{M}}(t) \cdot \left[P^3(t) + 3P^2(t)Q(t)\right], \tag{3}$$

где $P_{\text{М}}(t)$ — ВБР мажоритарного органа, $P(t) = e^{-\lambda t}$, $Q(t) = 1 - P(t)$.

4. Скользящее резервирование (ненагруженный резерв, экспоненциальный закон):

$$P_{\text{sys}}(t) = e^{-\lambda_0 t} \cdot \sum_{k=0}^{m_0} \frac{(\lambda_0 t)^k}{k!}, \tag{4}$$

где $\lambda_0 = \lambda \cdot n$ — интенсивность отказов нерезервированной системы, m_0 — число резервных элементов.

Важное замечание. При прочих равных условиях система с *ненагруженным* резервом надёжнее системы с *нагруженным*, но нагруженный резерв проще в реализации и не требует переключающих устройств. Сравнивая общее и раздельное резервирование, можно показать, что *поэлементное (раздельное) резервирование* всегда эффективнее общего при одинаковой суммарной избыточности [1].

3 Практическое задание

Задание включает системный анализ, аналитический расчёт и компьютерное моделирование.

3.1 Системный анализ и актуализация видов резервирования

Студенту необходимо сопоставить классические виды резервирования из Лекции 6 [1] с современными архитектурными решениями в распределённых системах. Заполните Таблицу 1 в отчёте, приведя для каждого вида резервирования не менее двух современных примеров реализации.

Таблица 1: Сопоставление видов резервирования с современными архитектурами

Вид резервирования	**Современные примеры реализации**
Общее нагруженное (постоянное)	?
Раздельное (поэлементное)	?
Замещением (ненагруженный резерв)	?
Скользящее	?
Мажоритарное (дробная кратность)	?
Комбинированный	?

Ориентир для анализа:

- **Мажоритарное резервирование** в современных системах — это не аппаратный «элемент голосования», а консенсус-алгоритмы.
- **Информационное резервирование** — это ECC-память DDR5.

3.2 Аналитический расчёт

Рассмотрим современный микросервисный кластер, состоящий из трёх функциональных блоков:

- **API Gateway** (2 экземпляра, $\lambda_1 = 5 \cdot 10^{-5}$ 1/ч),
- **Application Server** (3 экземпляра, $\lambda_2 = 10^{-4}$ 1/ч),

- **Database Tier** (PostgreSQL с Patroni, 3 узла, $\lambda_3 = 2 \cdot 10^{-5}$ 1/ч).

Рассчитайте $P_{\text{sys}}(t)$ за $t = 8\,760$ ч (1 год) для следующих схем.

- **Схема A.** Нерезервированная система (последовательное соединение всех 8 узлов).
- **Схема B.** Общее нагруженное резервирование API Gateway с кратностью $m = 1$ (дублирование), остальные блоки — без резервирования.
- **Схема C.** Раздельное резервирование: каждый блок имеет кратность m_i, равную числу его экземпляров минус один (т. е. API Gateway — $m_1 = 1$, App Server — $m_2 = 2$, DB Tier — $m_3 = 2$).
- **Схема D.** Мажоритарное резервирование для DB Tier по схеме «2 из 3» (с учётом ВБР мажоритарного органа $P_{\text{M}} = 0.9999$).
- **Схема E.** Скользящее резервирование для App Server: 3 основных + 2 резервных (ненагруженных) обработчика, любой из которых может заменить отказавшего.

Составьте сравнительную таблицу результатов и сделайте вывод о том, какая схема обеспечивает максимальную ВБР при одинаковой суммарной избыточности.

3.3 Статистическое моделирование на Python

Необходимо написать скрипт на Python, который методом Монте-Карло моделирует эксплуатацию резервированной системы и сравнивает эмпирическую ВБР с теоретической.

Требования к скрипту

- Реализовать функции расчёта теоретической ВБР для схем A – E по формулам (1) – (4).
- Провести $N = 10\,000$ испытаний для каждой схемы: генерировать времена отказов всех элементов (экспоненциальное распределение) и определять, не отказала ли система к моменту $t = 8\,760$ ч.
- Рассчитать эмпирическую ВБР $\widehat{P}_{\text{sys}}$ и доверительный интервал (95%).
- Построить график сравнения теоретических и эмпирических значений ВБР для всех пяти схем.
- Построить график зависимости $P_{\text{sys}}(t)$ от времени $t \in [0,\ 20\,000]$ ч для схем A, B, C (чтобы наглядно показать выигрыш резервирования).

Пример кода

```
"""
Лабораторная работа №4 по курсу "Надёжность информационных систем".
Структурное резервирование: аналитический расчёт и проверка методом
↪  Монте-Карло.
Схемы:
    A - нерезервированная (последовательная),
    B - общее нагруженное резервирование API Gateway (m=1),
```

```
    C - раздельное резервирование всех блоков,
    D - мажоритарное резервирование DB Tier (2 из 3),
    E - скользящее ненагруженное резервирование App Server.
"""

import math
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

# 1. Исходные данные
t_obs = 8760            # время наблюдения, часы (1 год)
N_sim = 10000           # число реализаций Монте-Карло
alpha = 0.05            # уровень значимости для доверительного интервала

# Блоки системы: (lambda_i, число основных элементов)
blocks = {
    'API Gateway'  : (5e-5,  2),
    'App Server'   : (1e-4,  3),
    'DB Tier'      : (2e-5,  3),
}

# ВБР мажоритарного органа (схема D)
P_majority_org = 0.9999

# 2. Теоретические функции ВБР для схем A-E
def P_single(lam, t):
    return math.exp(-lam * t)

def scheme_A(t):
    """Последовательное соединение всех 8 узлов."""
    lam_sys = sum(lam * n for lam, n in blocks.values())
    return math.exp(-lam_sys * t)

def scheme_B(t):
    """Общее нагруженное резервирование API Gateway (m=1), остальные без
    ↪ резерва."""
    lam_api, n_api = blocks['API Gateway']
    # Для нагруженного резерва с m=1 (дублирование): P = 1 - (1-P_elem)^2
    P_api_block = 1.0 - (1.0 - P_single(lam_api, t))**2
    lam_other = sum(lam * n for name, (lam, n) in blocks.items() if name !=
    ↪ 'API Gateway')
    return P_api_block * math.exp(-lam_other * t)

def scheme_C(t):
    """Раздельное резервирование: каждый блок имеет m_i = n_i - 1."""
    P_sys = 1.0
    for lam, n in blocks.values():
        m_i = n - 1
```

```
            # Формула общего резервирования для блока: P = 1 - (1 -
            ↪  e^-lt)^(m+1)
            P_block = 1.0 - (1.0 - P_single(lam, t))**(m_i + 1)
            P_sys *= P_block
        return P_sys

def scheme_D(t):
    """Мажоритарное резервирование DB Tier (2 из 3), остальные - как в схеме
    ↪  A."""
    lam_db, n_db = blocks['DB Tier']
    P_elem = P_single(lam_db, t)
    Q_elem = 1.0 - P_elem
    # ВБР мажоритарной группы: P^3 + 3*P^2*Q
    P_db_majority = P_elem**3 + 3 * P_elem**2 * Q_elem
    # Учитываем ВБР мажоритарного органа
    P_db = P_majority_org * P_db_majority
    # Остальные блоки - последовательно
    lam_other = sum(lam * n for name, (lam, n) in blocks.items() if name !=
    ↪  'DB Tier')
    return P_db * math.exp(-lam_other * t)

def scheme_E(t):
    """Скользящее ненагруженное резервирование App Server: 3 основных + 2
    ↪  резервных."""
    lam_app, n_app = blocks['App Server']
    lam_0 = lam_app * n_app          # интенсивность отказов
    ↪  нерезервированной группы
    m_0 = 2                          # число резервных элементов
    ↪  (ненагруженных)

    # Формула для скользящего ненагруженного резерва:
    # P(t) = exp(-lam_0 * t) * sum( (lam_0 * t)^k / k! ) for k=0 to m_0
    P_base = math.exp(-lam_0 * t)
    s = sum((lam_0 * t)**k / math.factorial(k) for k in range(m_0 + 1))
    P_app_sliding = P_base * s

    # Остальные блоки - последовательно
    lam_other = sum(lam * n for name, (lam, n) in blocks.items() if name !=
    ↪  'App Server')
    return P_app_sliding * math.exp(-lam_other * t)

schemes = {
    'A (нерезерв.)'      : scheme_A,
    'B (общее, API)'     : scheme_B,
    'C (раздельное)'     : scheme_C,
    'D (мажорит., DB)'   : scheme_D,
    'E (скользящ., App)' : scheme_E,
}

# 3. Аналитический расчёт P(t_obs)
```

```python
print(f"{'Схема':<22} {'P(8760 ч)':>12}")
print("-" * 36)
for name, func in schemes.items():
    P_val = func(t_obs)
    print(f"{name:<22} {P_val:>12.6f}")

# 4. Метод Монте-Карло
rng = np.random.default_rng(42)

def mc_scheme_A(t_obs, N_sim, rng):
    survived = np.ones(N_sim, dtype=bool)
    for lam, n in blocks.values():
        ttf = rng.exponential(scale=1.0/lam, size=(N_sim, n))
        survived = survived & (np.min(ttf, axis=1) >= t_obs)
    return np.mean(survived)

def mc_scheme_B(t_obs, N_sim, rng):
    survived = np.ones(N_sim, dtype=bool)
    lam_api, n_api = blocks['API Gateway']
    ttf_api = rng.exponential(scale=1.0/lam_api, size=(N_sim, n_api))
    survived = survived & (np.max(ttf_api, axis=1) >= t_obs)

    for name, (lam, n) in blocks.items():
        if name == 'API Gateway': continue
        ttf = rng.exponential(scale=1.0/lam, size=(N_sim, n))
        survived = survived & (np.min(ttf, axis=1) >= t_obs)
    return np.mean(survived)

def mc_scheme_C(t_obs, N_sim, rng):
    survived = np.ones(N_sim, dtype=bool)
    for lam, n in blocks.values():
        ttf = rng.exponential(scale=1.0/lam, size=(N_sim, n))
        survived = survived & (np.max(ttf, axis=1) >= t_obs)
    return np.mean(survived)

def mc_scheme_D(t_obs, N_sim, rng):
    survived = np.ones(N_sim, dtype=bool)
    lam_db, n_db = blocks['DB Tier']
    ttf_db = rng.exponential(scale=1.0/lam_db, size=(N_sim, n_db))
    db_survived_count = np.sum(ttf_db >= t_obs, axis=1)
    majority_org_survived = rng.random(N_sim) <= P_majority_org
    survived = survived & (db_survived_count >= 2) & majority_org_survived

    for name, (lam, n) in blocks.items():
        if name == 'DB Tier': continue
        ttf = rng.exponential(scale=1.0/lam, size=(N_sim, n))
        survived = survived & (np.min(ttf, axis=1) >= t_obs)
    return np.mean(survived)

def mc_scheme_E(t_obs, N_sim, rng):
```

```
    survived = np.ones(N_sim, dtype=bool)
    lam_app, n_app = blocks['App Server']
    lam_0 = lam_app * n_app
    m_0 = 2
    # Количество отказов за время t_obs распределено по Пуассону с
    ↪  параметром lam_0 * t_obs
    failures = rng.poisson(lam=lam_0 * t_obs, size=N_sim)
    survived = survived & (failures <= m_0)

    for name, (lam, n) in blocks.items():
        if name == 'App Server': continue
        ttf = rng.exponential(scale=1.0/lam, size=(N_sim, n))
        survived = survived & (np.min(ttf, axis=1) >= t_obs)
    return np.mean(survived)

mc_functions = {
    'A (нерезерв.)'      : mc_scheme_A,
    'B (общее, API)'     : mc_scheme_B,
    'C (раздельное)'     : mc_scheme_C,
    'D (мажорит., DB)'   : mc_scheme_D,
    'E (скользящ., App)' : mc_scheme_E,
}

print("\nМетод Монте-Карло:")
print(f"{'Схема':<22} {'P_hat':>10} {'95% ДИ':>22}")
print("-" * 56)

mc_results = {}
for name in schemes:
    P_hat = mc_functions[name](t_obs, N_sim, rng)
    # Доверительный интервал по нормальному приближению
    se = math.sqrt(P_hat * (1 - P_hat) / N_sim)
    z = stats.norm.ppf(1 - alpha / 2)
    ci_low = max(0.0, P_hat - z * se)
    ci_high = min(1.0, P_hat + z * se)
    mc_results[name] = P_hat
    print(f"{name:<22} {P_hat:>10.6f} [{ci_low:.4f}, {ci_high:.4f}]")

# 5. График: теория / Монте-Карло
names    = list(schemes.keys())
P_theory = [schemes[n](t_obs) for n in names]
P_mc     = [mc_results[n] for n in names]
errors   = [abs(P_theory[i] - P_mc[i]) / P_theory[i] * 100 if P_theory[i] >
↪  0 else 0.0 for i in range(len(names))]

x = np.arange(len(names))
width = 0.35

fig, ax1 = plt.subplots(figsize=(12, 6))
ax1.bar(x - width/2, P_theory, width, label='Теория', color='steelblue')
```

```
ax1.bar(x + width/2, P_mc, width, label='Монте-Карло', color='coral')
ax1.set_ylabel('Вероятность безотказной работы P(8760 ч)')
ax1.set_xticks(x)
ax1.set_xticklabels(names, rotation=15, ha='right')
ax1.legend(loc='upper left')
ax1.grid(True, alpha=0.3, axis='y')

ax2 = ax1.twinx()
ax2.plot(x, errors, 'ko-', markersize=8, label='Расхождение, %')
ax2.set_ylabel('Относительное расхождение, %', color='k')
ax2.legend(loc='upper right')

plt.title('Сравнение теоретической и эмпирической ВБР для схем A-E')
plt.tight_layout()
plt.savefig('lab04_theory_vs_mc.png', dpi=200)
print("\nГрафик сохранён в lab04_theory_vs_mc.png")
plt.show()

# 6. Динамика P_sys(t) для схем A, B, C
t_range = np.linspace(0, 20_000, 500)
P_A = [scheme_A(t) for t in t_range]
P_B = [scheme_B(t) for t in t_range]
P_C = [scheme_C(t) for t in t_range]

plt.figure(figsize=(10, 6))
plt.plot(t_range / 1e3, P_A, 'r-',  lw=2, label='A: нерезервированная')
plt.plot(t_range / 1e3, P_B, 'g--', lw=2, label='B: общее')
plt.plot(t_range / 1e3, P_C, 'b-.', lw=2, label='C: раздельное')
plt.axvline(t_obs / 1e3, color='k', ls=':', label='t = 1 год')
plt.xlabel('Время t, тыс. часов')
plt.ylabel('P_sys(t)')
plt.title('Динамика ВБР при различных схемах резервирования')
plt.legend()
plt.grid(True, alpha=0.5)
plt.tight_layout()
plt.savefig('lab04_Ps_dynamics.png', dpi=200)
print("График сохранён в lab04_Ps_dynamics.png")
plt.show()
```

4 Требования к отчёту

Отчёт должен содержать:

1. Титульный лист, цель работы и краткие теоретические положения.
2. Таблицу 1 с актуализированными примерами современных архитектурных решений для каждого вида резервирования (с указанием источников).
3. Результаты аналитических расчётов по схемам A – E с подробными выкладками и сравнительной таблицей.

4. Листинг программного кода с комментариями.

5. Графики, сгенерированные скриптом:

   - сопоставление теоретической и эмпирической (Монте-Карло) ВБР для всех схем с доверительными интервалами;
   - динамика $P_{\text{sys}}(t)$ для схем A, C, E на интервале $t \in [0,\, 20\,000]$ ч.

6. Анализ того, почему раздельное резервирование (схема C) эффективнее общего (схема B) при одинаковой суммарной избыточности.

7. Выводы о том, какие виды резервирования наиболее востребованы в современных облачных системах и почему.

5 Контрольные вопросы

1. Что такое структурное резервирование и в чём его отличие от информационного, временного и программного резервирования?

2. В чём принципиальное отличие постоянного (нагруженного) резервирования от резервирования замещением (ненагруженного)?

3. Почему система с ненагруженным резервом надёжнее системы с нагруженным при прочих равных условиях?

4. Что такое кратность резервирования? Чем отличается целая кратность от дробной?

5. Как устроено мажоритарное резервирование и какую роль в нём играет «элемент голосования»?

6. Чем раздельное резервирование отличается от общего, и почему оно эффективнее?

7. Как скользящее резервирование реализуется в современных системах очередей задач?

8. Почему резервирование на уровне компьютера (дублирование всей машины) экономически менее целесообразно, чем поэлементное резервирование?

9. Как изменится ВБР системы при переходе от нагруженного резерва к ненагруженному?

10. В каких случаях применение резервирования экономически не оправдано?

Список источников

Книги и учебные пособия

1. *Расулова С. С.* Надёжность информационных систем. — Ташкент : Ташкентский университет информационных технологий, 2007. — С. 118.

Актуальная документация

- Kubernetes Documentation: ReplicationController, ReplicaSet, Deployments.
- Patroni: A Template for PostgreSQL HA with ZooKeeper, etcd or Consul (https://patroni.readthedocs.io).
- The Raft Consensus Algorithm (https://raft.github.io).
- etcd: A Distributed Reliable Key-Value Store (https://etcd.io).
- Ceph Documentation (https://docs.ceph.com).