НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Кафедра обчислювальної техніки
(повна назва кафедри, циклової комісії)

РОЗРАХУНКОВО-ГРАФІЧНА РОБОТА

з дисципліни «Програмування вбудованих систем»

(назва дисципліни)

на тему: «Дослідження роботи планувальників роботи систем реального часу»

Студентки 3 курсу групи ІП-84
спеціальності
121 “Інженерія програмного забезпечення”
Романової В. А.
(прізвище та ініціали)
Керівник доцент Волокіта А. Н.

Київ — 2021

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки
(повна назва)

Кафедра обчислювальної техніки
(повна назва)

Освітньо-кваліфікаційний рівень бакалавр

Напрям підготовки 121 «Інженерія програмного забезпечення»
(шифр і назва)

З А В Д А Н Н Я

НА РОЗРАХУНКОВО-ГРАФІЧНУ РОБОТУ

Романовій Вікторії Андріївні
(прізвище, ім’я, по батькові)

1. Тема роботи «Дослідження роботи планувальників роботи систем реального часу»

керівник роботи Волокіта Артем Миколайович к.т.н., доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи 7 червня 2021 р.

3. Вхідні дані до роботи

- Алгоритм, кількість заявок, мінімальна вага заявки, максимальна вага заявки, інтенсивність
- мови (бібліотеки) програмування: Python3 (numpy, matplotlib)
- середовище розробки: PyCharm

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

- огляд алгоритмів планувальника
- розробка і тестування програми

5. Перелік графічного матеріалу

- скріншоти програми

7. Дата видачі завдання 29.04.2021

ЗМІСТ

Основні теоретичні відомості

Планування виконання завдань (англ. Scheduling) є однією з ключових концепцій в багатозадачності і багатопроцесорних систем, як в операційних системах загального призначення, так і в операційних системах реального часу. Планування полягає в призначенні пріоритетів процесам в черзі з пріоритетами.

Найважливішою метою планування завдань є якнайповніше завантаження доступних ресурсів. Для забезпечення продуктивності системи планувальник має опиратися на:

- Використання процесора(-ів) — дати завдання процесору, якщо це можливо.
- Пропускна здатність — кількість процесів, що виконуються за одиницю часу.
- Час на завдання — кількість часу, для повного виконання певного процесу.
- Очікування — кількість часу, який процес очікує в черзі готових.
- Час відповіді — час, який проходить від подання запиту до першої відповіді на запит.
- Справедливість — Рівність процесорного часу для кожної нитi

У середовищах обчислень реального часу, наприклад, на пристроях, призначених для автоматичного управління в промисловості (наприклад, робототехніка), планувальник завдань повинен забезпечити виконання процесів в перебігу заданих часових проміжків (час відгуку); це критично для підтримки коректної роботи системи реального часу.

Система масового обслуговування (СМО) — система, яка виконує обслуговування вимог (заявок), що надходять до неї. Обслуговування вимог у СМО проводиться обслуговуючими приладами. Класична СМО містить від одного до нескінченного числа приладів. В залежності від наявності можливості очікування вхідними вимогами початку обслуговування СМО (наявності черг) поділяються на:

1) системи з втратами, в яких вимоги, що не знайшли в момент надходження жодного вільного приладу, втрачаються;

2) системи з очікуванням, в яких є накопичувач нескінченної ємності для буферизації надійшли вимог, при цьому очікують вимогу утворюють чергу;

3) системи з накопичувачем кінцевої ємності (чеканням і обмеженнями), в яких довжина черги не може перевищувати ємності накопичувача; при цьому вимога, що надходить в переповнену СМО (відсутні вільні місця для очікування), втрачається.

Основні поняття СМО:

- *Вимога (заявка)* — запит на обслуговування.
- *Вхідний потік вимог* — сукупність вимог, що надходять у СМО.
- *Час обслуговування* - період часу, протягом якого обслуговується вимогу.

Вимоги до системи

Вхідні задачі

Вхідними заявками є обчислення, які проводилися в лабораторних роботах 1-3, а саме обчислення математичного очікування, дисперсії, автокореляції, перетворення Фур'є.

Вхідні заявки характеризуються наступними параметрами:

1) час приходу в систему – Tp – потік заявок є потоком Пуассона або потоком Ерланга k-го порядку;

2) час виконання (обробки) – To; математичним очікуванням часу виконання є середнє значення часу виконання відповідних обчислень в попередніх лабораторних роботах;

3) крайній строк завершення (дедлайн) – Td –задається (випадково?); якщо заявка залишається необробленою в момент часу t = Td, то її обробка припиняється і вона покидає систему.

Потік вхідних задач

Потоком Пуассона є послідовність випадкових подій, середнє значення інтервалів між настанням яких є сталою величиною, що дорівнює $1/\lambda$, де λ – інтенсивність потоку [3].

Потоком Ерланга k-го порядку називається потік, який отримується з потоку Пуассона шляхом збереження кожної (k+1)-ї події (решта відкидаються). Наприклад, якщо зобразити на часовій осі потік Пуассона, поставивши у відповідність кожній події деяку точку, і відкинути з потоку кожну другу подію (точку на осі), то отримаємо потік Ерланга 2-го порядку. Залишивши лише кожну третю точку і відкинувши дві проміжні, отримаємо потік Ерланга 3-го порядку і т.д. Очевидно, що потоком Ерланга 0-го порядку є потік Пуассона. Також легко

бачити, що зі збільшенням порядку потік Ерланга прямує до рівномірного розподілу.

Спорадичними задачами називають неперіодичні задачі, у яких коефіцієнт дедлайну близький до 1 (задачі повинні бути майже одразу виконані інакше дедлайн буде порушено).

Приорітети заявок

Заявки можуть мати пріоритети – явно задані, або обчислені системою (в залежності від алгоритму обслуговування або реалізації це може бути час обслуговування (обчислення), час до дедлайну і т.д.). Заявки в чергах сортуються за пріоритетом. Є два види обробки пріоритетів заявок:

1) без витіснення – якщо в чергу до ресурсу потрапляє заявка з більшим пріоритетом, ніж та, що в даний момент часу обробляється ним, то вона чекає завершення обробки ресурсом його задачі.

2) з витісненням – якщо в чергу до ресурсу потрапляє заявка з більшим пріоритетом, ніж та, що в даний момент часу обробляється ним, то вона витісняє її з обробки; витіснена задача стає в чергу.

У даній роботі всі алгоритми реалізовано із двома чергами задач: загальною чергою задач та захищеною чергою для приорітетних задач. Задачі із захищеної черги можуть витісняти звичайні задачі та завжди обслуговуються в першу чергу. Задачі зі звичайної черги можуть надійти на виконання лише після того як спорожнилася черга захищених задач.

Дисципліна обслуговування

Вибір заявки з черги на обслуговування здійснюється за допомогою так званої дисципліни обслуговування. Їх прикладами є FIFO (прийшов першим - обслуговується першим), LIFO (прийшов останнім - обслуговується першим),

RANDOM (випадковий вибір). У системах з очікуванням накопичувач в загальному випадку може мати складну структуру.

Дисципліна FIFO

Алгоритм FIFO (first in, first out - перший прийшов, перший вийшов) є одним із найпростіших алгоритмів планування і полягає у виборі заявок у мірі їх надходження. Під час надходження заявок алгоритм ігнорує всі параметри, крім часу надходження. Щоразу вибирається та заявка, яка прийшла раніше. Через надмірну простоту та ігнорування особливостей заявок, FIFO мало використовується. Належить до статичних алгоритмів планування.

Дисципліна RM

Алгоритм RM (Rate Monotonic) належить до динамічних алгоритмів планування із фіксованими пріоритетами. Планувальник визначає найбільш пріоритетними ті заявки, час виконання яких є найменшим. Динаміка планування полягає у тому, що надходження нової заявки провокує повну переоцінку всієї черги на виконання. Таким чином досягається зменшення накопиченої черги заявок, однак можливе порушення дедлайнів більш довгих заявок, якщо черга надто забивається малими заявками. [1]

Дисципліна EDF

Алгоритм EDF (Earliest Deadline First - найранніший дедлайн перший) належить до динамічних алгоритмів планування із динамічним пріоритетом. Планувальник визначає найбільш пріоритетними ті заявки, дедлайн яких є найближчим. Динаміка планування полягає у тому, що надходження нової заявки провокує повну переоцінку всієї черги на виконання. Таким чином досягається максимальне виконання дедлайнів, але може збільшуватися час очікування заявки в системі (менш термінові заявки посуваються). [2]

Розробка програми

Мова програмування: Python3, середовище: PyCharm.

Було використано бібліотеку NumPy заради потоку Пуассона та генерації послідовності типу float, а також matplotlib.plt для візуалізації графіків.

```
class Task:
    """
    Клас, в якому зберігаються параметри заявки
    :param time_of_arrival: час прибуття в систему планування
    :param time_of_execution: час виконання заяви
    :param k: коефіцієнт для розрахунку коректного дедлайна
    """


def generate_stream(size, lam, time_of_execution, deadline):
    """
    Функція, яка генерує потік тасків певного типу за допомогою потоку Пуассона
    :param size: орієнтовна кількість тактів
    :param lam: лямбда для потоку Пуассона
    :param time_of_execution: для налаштування тасків
    :param deadline: для налаштування тасків
    :return: масив тасків
    """


class Processor:
    """
    Клас, в якому зберігається стан процесора
    """
    def __init__(self):
        self.status = 1  # 0 - busy, 1 - free
        self.task = None
```

Клас Процесор має наступні методи:
set_task(self, task), що встановлює новий таск для обробки
work(self), що накопичує прогрес виконання таска та перевіряє його завершеність
clear(self), що очищає стан процесора

```
class Scheduler:
    """
    Клас, що симулює алгоритми планування
    :param size: кількість тактів системи
```

```
    :param lam: лямбда, необхідна для генерації потоку заявок
    :param processors: кількість доступних процесорів
    """
```

Клас Шедулер має наступні методи:

check_stream(stream, buffer, tact, method): перевіряє чи надійшов новий таск та генерує нову чергу вхідних заявок згідно з методом (дисципліною)

check_downtime(buffer, processors): перевіряє чи є при порожньому буфері хоч один вільний процесор

clear_processors(self): скидає статус усіх наявних процесорів

scheduling(self, method=None): здійснює планування згідно з дисципліною

```
    def fifo(self):
        return self.scheduling()

    def rm(self):
        return self.scheduling('rm')

    def edf(self):
        return self.scheduling('edf')
```

show(self): виводить у консоль корисні значення значення

```
class Graphics:
    """
    Клас для побудови графіків для усіх дисциплін планування
    :param size: кількість тактів системи планувальння
    :param intensity: інтенсивність потоку для першого графіку
    :param processors: кількість доступних процесорів для системи планування
    """
```

Клас Графіки містить наступні методи:

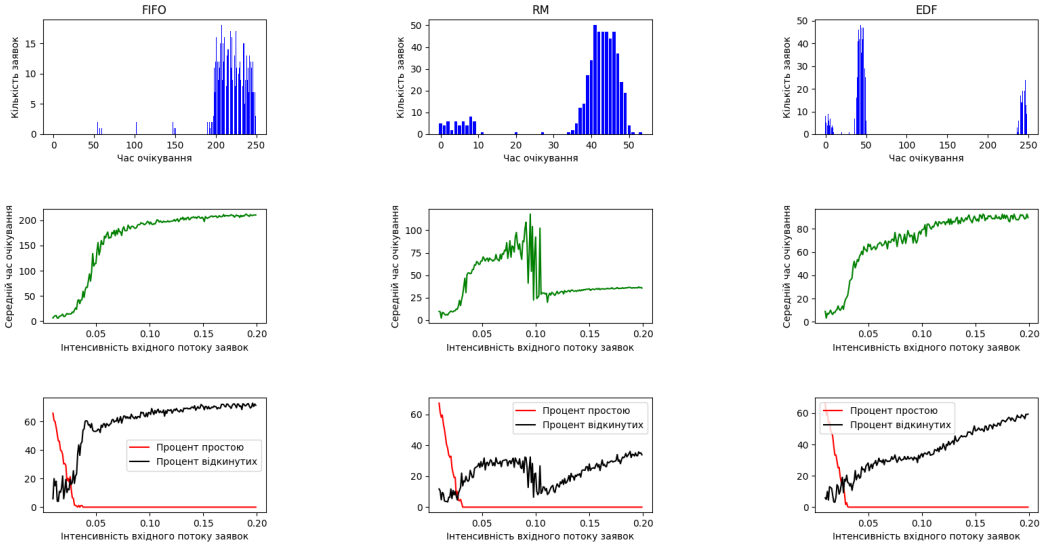
show(self): генерує розмітку для майбутніх графіків та заповнює її

bar(self, ax, method, title): заповнює вказане місце стовпчастою діаграмою, що відображає “Графік залежності кількості заявок від часу очікування при фіксованій інтенсивності вхідного потоку заявок.”

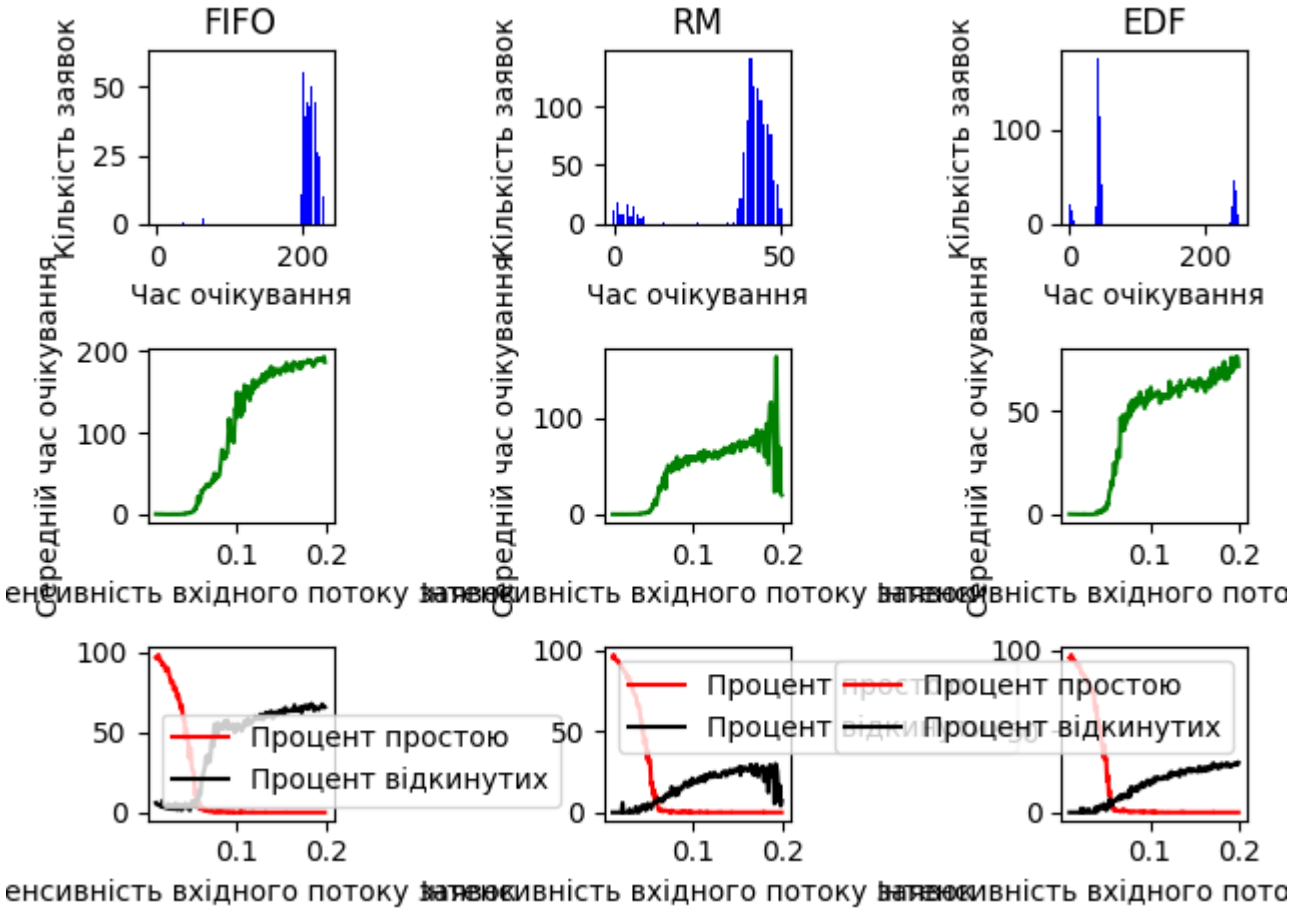
plots(self, ax1, ax2, method): заповнює вказані місця графіками “Графік залежності середнього часу очікування від інтенсивності вхідного потоку заявок.” та “Графік залежності проценту простою ресурсу та відсотку прострочених заявок від інтенсивності вхідного потоку заявок.”

Результати виконання

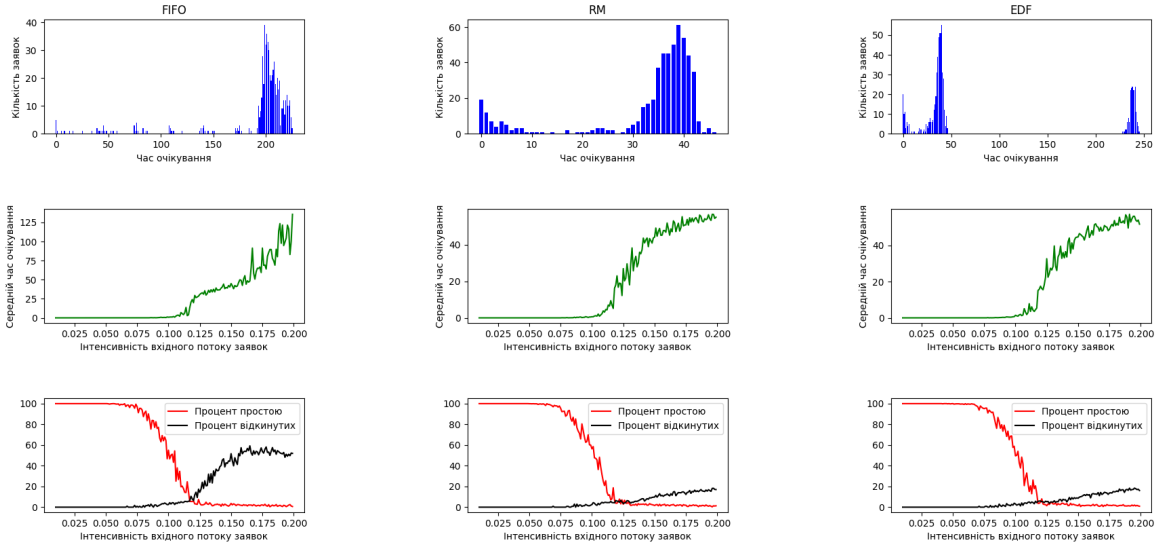
https://github.com/V1ckeyR/RTS/tree/main/rgr



Графіки для 1 процесора



Графіки для 2х процесорів



Графіки для 4х процесорів

```
------------------------------FIFO------------------------------
Інтенсивність: 1.0
Кількість процесорів: 2
Середній розмір вхідної черги заявок: 303.143
Середній час очікування заявки в черзі: 208.91
Кількість прострочених заявок: 1202
Відношення кількості прострочених заявок до загальної кількості: 0.75
------------------------------RM------------------------------
Інтенсивність: 1.0
Кількість процесорів: 2
Середній розмір вхідної черги заявок: 286.971
Середній час очікування заявки в черзі: 39.1
Кількість прострочених заявок: 853
Відношення кількості прострочених заявок до загальної кількості: 0.54
------------------------------EDF------------------------------
Інтенсивність: 1.0
Кількість процесорів: 2
Середній розмір вхідної черги заявок: 148.905
Середній час очікування заявки в черзі: 93.58
Кількість прострочених заявок: 1218
Відношення кількості прострочених заявок до загальної кількості: 0.78
```

```
----------------------------FIFO----------------------------
Інтенсивність: 0.5
Кількість процесорів: 2
Середній розмір вхідної черги заявок: 147.704
Середній час очікування заявки в черзі: 207.86
Кількість прострочених заявок: 588
Відношення кількості прострочених заявок до загальної кількості: 0.75
----------------------------RM----------------------------
Інтенсивність: 0.5
Кількість процесорів: 2
Середній розмір вхідної черги заявок: 140.176
Середній час очікування заявки в черзі: 37.24
Кількість прострочених заявок: 307
Відношення кількості прострочених заявок до загальної кількості: 0.4
----------------------------EDF----------------------------
Інтенсивність: 0.5
Кількість процесорів: 2
Середній розмір вхідної черги заявок: 75.793
Середній час очікування заявки в черзі: 91.99
Кількість прострочених заявок: 543
Відношення кількості прострочених заявок до загальної кількості: 0.67
```

Використана література

1. Rate-monotonic scheduling [Електронний ресурс] // Wikipedia, The Free Encyclopedia – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Rate-monotonic_scheduling.
2. Earliest deadline first scheduling [Електронний ресурс] // Wikipedia, The Free Encyclopedia – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Earliest_deadline_first_scheduling.
3. Пуассонівський процес [Електронний ресурс] // Вікіпедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9F%D1%83%D0%B0%D1%81%D1%81%D0%BE%D0%BD%D1%96%D0%B2%D1%81%D1%8C%D0%BA%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81.

ДОДАТОК

Вихідний код: https://github.com/V1ckeyR/RTS/tree/main/rgr