МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп’ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

Вишневський Дмитро Сергійович

Метод зниження пікових температур в IoT пристроях за допомогою міграції та заміни завдань / A method for reduce peak temperatures in IoT devices by migrating and replacing tasks

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп’ютерні науки

Кваліфікаційна робота

Виконав студент групи
КНм-21
Д. С. Вишневський
_____

Науковий керівник:
к.т.н., доцент
О.Р. Осолінський
_____

Кваліфікаційну роботу
допущено до захисту:
«___» _____ 20___ р.
Завідувач кафедри
_____ М.П. Комар

ТЕРНОПІЛЬ - 2023

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «магістр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Вишневському Дмитру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Метод зниження пікових температур в IoT пристроях за допомогою міграції та заміни завдань / A method for reduce peak temperatures in IoT devices by migrating and replacing tasks
керівник роботи к.т.н., доцент О.Р. Осолінський
затверджені наказом по університету від 8 грудня 2022 року № 491.

2. Строк подання студентом закінченої кваліфікаційної роботи 1 грудня 2023 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити
- дослідити методи планування завдань для IoT пристроїв;
- дослідити типи базові принципи розсіювання потужності мікропроцесорних та мікроконтролерних системах;
- дослідити методи планування з урахуванням температури;
- дослідити методи теплового моделювання в багатоядерних системах;
- дослідити незалежні та залежні задачі на багатоядерних IoT пристроях
- розробити метод зниження пікових температур в іот пристроях за допомогою міграції та заміни завдань;
- розробити алгоритм планування з урахуванням теплового режиму на основі динамічної теплової моделі;
- розробити та реалізувати на основі методу зниження пікових температур в IoT пристроях за допомогою міграції та заміни завдань розробити модель оцінки методів управління ресурсами та провести оцінку виконання моделі;

5. Перелік графічного матеріалу у роботі
- Блок-схема алгоритму розподілу завдань;

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 8 грудня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Огляд методів планування завдань для іот пристроїв, що враховують енергію та температуру	12.2022 р. – 03.2023 р.	
2	Метод зниження пікових температур в іот пристроях за допомогою міграції та заміни завдань	03.2023 р. – 05.2023 р.	
3	Реалізація методу зниження пікових температур в іот пристроях за допомогою міграції та заміни завдань	05.2023 р. – 11.2023 р.	
4	Повне завершення та представлення кваліфікаційної роботи на кафедру	01.12.2023 р.	

Студент _____ Д. С. Вишневський
підпис

Керівник роботи _____ к.т.н., доцент О.Р. Осолінський
підпис

РЕЗЮМЕ

Кваліфікаційна робота на тему «Метод зниження пікових температур в IoT пристроях за допомогою міграції та заміни завдань» на здобуття освітнього ступеня «Магістр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 85 сторінок і містить 25 ілюстрацій, 3 таблиці, 2 додатки та 51 використаних джерел.

Метою даної кваліфікаційної роботи є зниження пікових температур в іот пристроях за допомогою міграції та заміни завдань.

Методи досліджень: теорія електричних і магнітних кіл, системний аналіз, математична статистика, методи комп'ютерного моделювання.

Результати дослідження: вдосконалено методу зниження пікових температур в іот пристроях за допомогою міграції та заміни завдань за рахунок планування з урахуванням теплового режиму на основі динамічної теплової моделі, що дозволило зманшити перегрівання ядер процесора модуля ARM без суттєвих втрат в продуктивності.

Результати роботи можуть успішно застосовуватися для використання в індустрії 4.0 та Інтернету речей в додатках де є проблема оптимізації температурних режимів в мікрокропроцесорних системах та IoT.

Ключові слова: IoT, ПЛАНУВАЛЬНИК ЗАВДАНЬ, ТЕПЛОВЕ МОДЕЛЮВАННЯ, ПЛАНУВАННЯ, ТЕПЛОВИЙ РЕЖИМ, БАГАТОЯДЕРНІ СИСТЕМИ.

ABSTRACT

Qualification work on the topic «A method for reduce peak temperatures in IoT devices by migrating and replacing tasks» for Master's degree on speciality 122 «Computer Science» educational and professional program «Computer Science» is written on 85 pages and it contains 25 figures, 3 tablei, 2 annexes and 51 sources.

The purpose of this qualification work is to reduce peak temperatures in IoT devices by migrating and replacing tasks.

Research methods: theory of electric and magnetic circuits, system analysis, mathematical statistics, computer modeling methods.

Research results: improved the method of reducing peak temperatures in IoT devices by migrating and replacing tasks through thermal scheduling based on a dynamic thermal model, which allowed to prevent overheating of the ARM module processor cores without significant performance losses.

The results of the work can be successfully applied for use in industry 4.0 and the Internet of Things in applications where there is a problem of optimizing temperature conditions in microcomputer systems and IoT.

Keywords: IoT, TASK SCHEDULER, THERMAL MODELING, SCHEDULING, THERMAL MODE, MULTI-CORE SYSTEMS.

ЗМІСТ

ВСТУП

Методи зниження пікових температур у пристроях Інтернету речей (IoT) відіграють критичну роль у забезпеченні стабільної та ефективної роботи цих пристроїв. Одним із перспективних підходів є використання стратегій міграції та заміни завдань.

Методи зниження пікових температур у IoT пристроях набувають все більшої важливості у зв'язку з потребою в удосконаленні та оптимізації їхньої роботи. Серед найбільш перспективних підходів - використання стратегій міграції та заміни завдань, які дозволяють управляти навантаженням пристроїв та розподіляти його для зниження теплових піків.

Ця проблематика стає актуальною через ріст функціональності та обчислювальних можливостей IoT пристроїв, що призводить до збільшення тепловиділення та підвищення ризику перегріву. Дослідження у цьому напрямку відкривають шляхи для розробки нових алгоритмів управління та оптимізації енергоспоживання, сприяючи підвищенню стабільності, надійності та тривалості роботи Інтернету речей.

Метою роботи є розробка методу зниження пікових температур в IoT пристроях за допомогою міграції та заміни завдань.

Досягнення цієї мети зумовило потребу теоретичних розробок, визначення та послідовного вирішення таких завдань:

1. дослідити методи планування завдань для IoT пристроїв;
2. дослідити типи базові принципи розсіювання потужності мікропроцесорних та мікроконтролерних системах;
3. дослідити методи планування з урахуванням температури;
4. дослідити методи теплового моделювання в багатоядерних системах;
5. дослідити незалежні та залежні задачі на багатоядерних IoT пристроях;
6. розробити метод зниження пікових температур в іот пристроях за допомогою міграції та заміни завдань;

7. розробити алгоритм планування з урахуванням теплового режиму на основі динамічної теплової моделі;
8. розробити та реалізувати на основі методу зниження пікових температур в IoT пристроях за допомогою міграції та заміни завдань розробити модель оцінки методів управління ресурсами та провести оцінку виконання моделі;

Об'єктом дослідження є багатоядерні процесори в складі контролерів для IoT модулів.

Предметом дослідження є методи зниження пікових температур IoT пристрою за рахунок міграції завдань.

Методи дослідження. У роботі використані методи математичного і графічного програмування; системного аналізу; моделювання.

Наукова новизна отриманих результатів полягає у розробці методу зниження пікових температур в IoT пристроях за допомогою міграції та заміни завдань.

Практичне значення одержаних результатів полягає у створені алгоритму та правил для планувальника завдань IoT пристрою з урахуванням теплового режиму на основі динамічної теплової моделі.

Структура та обсяг роботи. Дипломна робота складається із вступу, трьох розділів, висновків, списку використаних джерел. Повний обсяг роботи становить 85 ст. комп'ютерного тексту, який включає 25 рисунків. Список використаних джерел із 51 найменувань викладено на 6 сторінках.

Апробація результатів дослідження. Основні теоретичні положення роботи й практичні результати дослідження доповідалися й обговорювалися на Міжнародній науковій інтернет-конференції, (м. Тернопіль, Україна, м. Ополе, Польща, 9-10 листопада 2023 р.), «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення» та на Всеукраїнській студентській науковійконференції, «Розвиток сучасної науки: актуальні питання теорії та практики» м. Львів, 17 листопада, 2023 рік.

1 ОГЛЯД МЕТОДІВ ПЛАНУВАННЯ ЗАВДАНЬ ДЛЯ ІОТ ПРИСТРОЇВ, ЩО ВРАХОВУЮТЬ ЕНЕРГІЮ ТА ТЕМПЕРАТУРУ

1.2 Базові принципи розсіювання потужності мікропроцесорних та мікроконтролерних системах

CMOS-пристрої є основою більшості сучасних обчислювальних систем загального призначення та IoT пристроїв в тому числі. Потужність, що споживається в КМОП схемах, можна розділити на три складові: динамічну, статичну і потужність короткого замикання. Протягом багатьох років динамічне розсіювання потужності, яке відбувається через перемикання транзисторів, було домінуючим фактором в енергоспоживанні КМОП. Динамічну потужність можна апроксимувати наступною формулою:

$$P_d = C_l \cdot N_{sw} \cdot V_{dd}^2 \cdot f \quad (1)$$

де C_l - ємність навантаження, значна частина якої пов'язана з проводами, N_{sw} - середня кількість перемикань схеми за такт, V_{dd} - напруга живлення і f - тактова частота.

Зменшення розмірів елементів в останніх технологічних поколіннях дозволило суттєво знизити напругу живлення (Vdd), що в чотири рази зменшує розсіювану потужність.

На жаль, зменшення напруги живлення збільшує затримку схеми, і тому схеми не можуть працювати на тій самій тактовій частоті. Точніше кажучи, тактова частота майже лінійно залежить від напруги живлення наступним:

$$f = k \cdot \frac{(V_{dd} - V_{th})^2}{V_{dd}} \quad (2)$$

де k - константа, а V_{th} - порогова напруга [1], [2]. У свою чергу, P_d приблизно кубічно пов'язана з f: $P_d \approx C_l \cdot N_{sw} \cdot \frac{f^3}{k^2}$. Як наслідок, зменшення напруги живлення і тактової частоти зменшує динамічну потужність у кубічному відношенні (і динамічну енергію в квадратичному) за рахунок майже лінійного збільшення часу виконання.

Другий компонент розсіюваної потужності, статична (або потужність витоку), пов'язаний з існуванням струму витоку, що протікає між джерелом живлення і землею. Підпороговий витік, основний підкомпонент потужності витоку, являє собою потужність, що розсіюється транзистором, затвор якого повинен бути закритий.

В останніх технологічних поколіннях потужність підпорогового витоку зросла в геометричній прогресії порівняно з попередніми поколіннями, головним чином через спільне зниження напруги живлення і порогової напруги.

В даний час потужність витоку становить 20-40% від загальної потужності розсіювання [3]. На особливу увагу заслуговує експоненціальна залежність потужності витоку від температури. З підвищенням температури системи потужність витоку стає більш значною частиною загальної потужності.

На особливу увагу заслуговує експоненціальна залежність потужності витоку від температури. З підвищенням температури системи потужність витоку стає більш значною частиною загальної потужності. Більшість досліджень, спрямованих на зменшення потужності витоку, проводяться на технологічному рівні [4], [5] та на мікроархітектурному рівні [6], [7], [8], і тому вони є складною задачею для науковців які напряму не дотичні до засобів виробництва мікропроцесорних систем та IoT пристроїв також.

Також, потужність короткого замикання або "глюка" - це потужність, що розсіюється при одночасній провідності n і p транзисторів КМОП-затвора. Оскільки потужність короткого замикання порівняно незначна порівняно з динамічною і статичною потужністю, ця складова енергоспоживання не привертає стільки уваги, скільки інші, як з боку дослідницької спільноти, так і з боку промисловості.

1.2.1 Керування енергоспоживанням під час виконання

Вивчення взаємозв'язків у рівнянні (1) обґрунтовує три основні напрямки досліджень, спрямованих на зменшення споживання енергії системою за допомогою програмних методів, що використовуються під час виконання.

Перша група рішень базується на динамічному масштабуванні напруги та частоти (DVFS) та динамічному управлінні живленням (DPM). Ці рішення динамічно змінюють напругу та частоту мікропроцесорної системи або IoT модуля. Оскільки зниження напруги і частоти може призвести до зниження продуктивності, рішення на основі DVFS/DPM враховують взаємозв'язок між частотою і продуктивністю, беручи до уваги той факт, що деякі програми сповільнюються менше, ніж інші, при зниженні частоти.

Друга група рішень, Thermal Management, пов'язана з тим, що температура мікроконтролера або пристрою IoT має значний вплив на енергоспоживання.

Наприклад, коли процесор в середині модуля нагрівається, система повинна збільшити активність інфраструктури охолодження (наприклад, вентиляторів), якщо така передбачена. Ця група рішень ґрунтується на спостереженні, що різні додатки по-різному впливають на температуру мікропроцесорної системи, і використовує цікаві просторові залежності, коли йдеться про контроль температури.

Наприклад, запуск двох потоків, що генерують тепло, на сусідніх ядрах створить більший температурний градієнт, ніж запуск цих потоків на окремих частинах чіпа. Таким чином, алгоритми керування температурою виконують просторове та часове розміщення потоків таким чином, щоб мінімізувати теплові аварійні ситуації та зменшити теплові градієнти, які виникають, коли варіація температури на різних частинах чіпа є високою.

Третя група рішень, Asymmetry-Aware Scheduling, пов'язана з системами, які є асиметричними за своєю природою. По суті, енергоспоживання системи залежить від її мікроархітектури: наприклад, системи, що працюють на вищих частотах і мають складніші конвеєри та більші кеші, споживають більше енергії, ніж менші за розміром IoT модулі з простішими функціями.

Однак простіші та менші за розміром мікропроцесорні системи також мають нижчу продуктивність. Щоб вирішити проблему компромісу між енергоспоживанням та продуктивністю, деякі дослідники запропонували будувати системи, які включають як складні, швидкі та енергоємні ядра, так і прості, економні та енергоефективні ядра. Такі системи називаються асиметричними.

У симетричних системах ISA до яки входить більшість контролерів IoT з асиметричною продуктивністю тягар прийняття рішення про те, який тип ядра слід використовувати для запуску певного потоку, покладається на планувальник операційної системи, тому інтелектуальні алгоритми планування, які оптимізують енергоспоживання та продуктивність системи, мають вирішальне значення.

Інша категорія асиметричних систем - це системи, в яких асиметрія не є явною конструктивною особливістю, а виникає через рутинні апаратні несправності (наприклад, ядра, які мали б працювати на однакових частотах, насправді працюють на різних). Ці системи стикаються з тими ж проблемами, що і явно асиметричні системи. Однак через динамічну природу їхньої асиметрії (наприклад, апаратні несправності, що роблять систему асиметричною, можуть виникнути в будь-який момент під час виконання).

Підсумовуючи, оптимізацію управління енергоспоживанням у планувальнику потоків можна розділити на три категорії відповідно до механізму, на який вони спираються:

(1) динамічне керування живленням і динамічне масштабування вольтажу/частоти,

(2) керування тепловим режимом

(3) планування з урахуванням асиметрії.

У Таблиці 1 наведено основні проблеми, на які спрямовані ці рішення, а також ключові ідеї, що лежать в їх основі.

Таблиця 1.1 - Короткий огляд методів керування мікропроцесорними пристроями живленням під час виконання програм

Метод	Цілі	Основна ідея	Підходи до вирішення проблеми
DVFS/DPM	Зменшити робочу напругу/частоту або стан живлення мінімізуючи або обмежуючи вплив на продуктивність.	Деякі програми зазнають менших втрат продуктивності від зниження частоти, ніж інші. Додатки, що працюють у режимі реального часу та мають дедлайни, мають низьку продуктивність: ніяких переваг від підвищення продуктивності після дотримання дедлайну.	Частота процесора IoT модуля вибирається на основі 1) моделі продуктивності для взаємозв'язку між продуктивністю та енергоспоживанням на основі характеристик програми; 2) статичної моделі для конкретного процесора
Температурний менеджмент	Зменшення теплових аварійних ситуації та температурні градієнти, активно або реактивно переміщуючи "гарячі" потоки подалі від ділянок, які можуть перегрітися.	Тип обчислень, які виконують потоки, впливає на те, скільки тепла вони будуть генерувати.	1) переміщення потоків з гарячих ділянок у холодні; 2) Проактивно уникання об'єднання в кластери тих потоків, які з більшою ймовірністю виділяють тепло
Планування з урахуванням асиметрії	Призначайте потоки на ядра так, щоб максимізувати продуктивність	Відносна перевага роботи на "швидкому" чи "повільному" ядрі залежить від типу виконуваного коду: наприклад, обчислювально інтенсивний чи	1) Метод спроб і помилок: планування потоків, вимірювання продуктивності, коригування; 2) Аналітичне моделювання:

		пам'яттєво інтенсивний, паралельний чи послідовний.	прогнозування продуктивності потоку на тому чи іншому типі ядра, враховуючи характеристики потоків, які визначені в Інтернеті.

1.2.2 Динамічне масштабування напруги/частоти та динамічне керування живленням

Динамічне керування живленням (DPM) використовує той факт, що певні електронні компоненти/пристрої можуть залишатися бездіяльними значний проміжок часу, тому їх можна вимикати під час неактивних періодів для економії енергії. Виробники апаратного та програмного забезпечення домовилися про створення стандартів, таких як ACPI (Advanced Configu ration and Power Interface) [9], які передбачають кілька режимів роботи, що дозволяють вимикати окремі частини системи, такі як ядра процесора, периферійні пристрої, порти і.т.д.

Сучасні пристрої пропонують різні режими живлення. Як ілюстративний приклад, на рисунку 1 зображено різні режими живлення для процесорів, що реалізують специфікацію ACPI. Ідея DPM полягає в тому, що ці пристрої періодично переходять у режим очікування, коли немає ніякої роботи, і в результаті можуть бути переведені в менш енергоємний сплячий стан.

Недоліком є те, що коли робота стає доступною, пристрій повинен бути переведений назад в робочий стан, і цей перехід спричиняє додаткову затримку. Чим нижчий стан енергоспоживання (більше число, пов'язане з ним), тим більша економія енергії; однак затримка переходу також стає вищою для більш низьких активних станів, це особливо стосується IoT пристроїв.

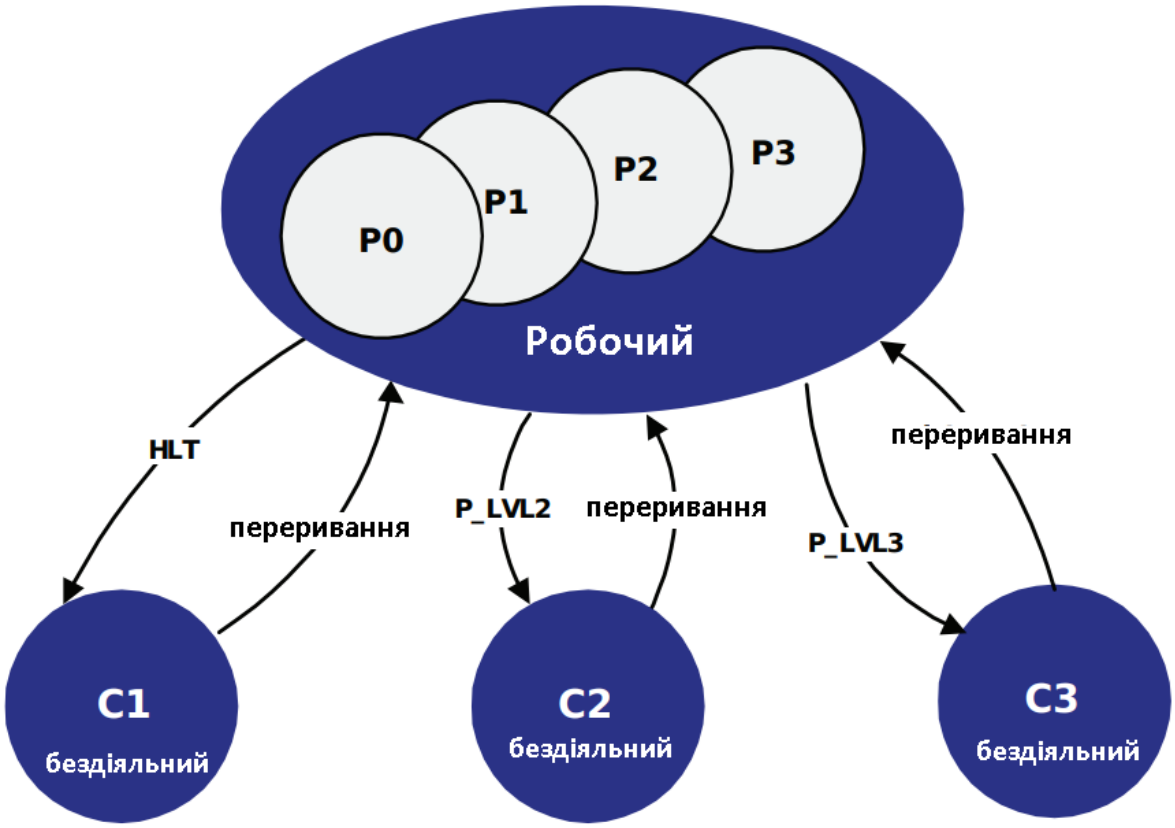


Рисунок 1.1 – Стани процесора модуля IoT

Процесор може перебувати або в неактивному стані з дуже низьким енергоспоживанням (C1, C2, ... , Ck), або в активному стані (C0), коли він може виконувати інструкції. У свою чергу, для процесорів у стані C0 компроміс між продуктивністю та енергоспоживанням може бути досягнутий за допомогою декількох станів продуктивності (Px), які зазвичай характеризуються різними тактовими частотами та напругами живлення.

У той час як DPM переводить компоненти в неактивні, але енергоефективні стани, динамічне масштабування напруги і частоти (DVFS) знижує динамічну потужність, що використовується процесором, обмежуючи його тактову частоту (f) і/або напругу живлення (Vdd). Як зазначалося раніше, спільне зменшення f та Vdd дозволяє зменшити динамічну потужність у кубічному та квадратичному відношенні, але може призвести до збільшення часу виконання.

1.2.3 Алгоритми з використанням DVFS та DPM

Дослідження в роботі [10] були першими, хто запропонував використовувати DVFS для зменшення споживання енергії в обчислювальних системах. Було досліджено набір алгоритмів планування на рівні ОС для зменшення часу простою процесора шляхом динамічного регулювання рівня DVFS та оцінено кожен алгоритм за допомогою змодельованих трас виконання.

Перші дослідження, що використовували DVFS для зменшення енергоспоживання, з'явилися в контексті систем реального часу, де дедлайни завдань задані, а їхній найгірший час виконання (WCET), як правило, відомий заздалегідь.

Багато алгоритмів, запропонованих у цій галузі, використовують недоліки продуктивності, наявні в додатках реального часу [11], [12], [13], [14].

Загальний підхід полягає в тому, щоб мінімізувати споживання енергії за рахунок зниження продуктивності процесора (рівень DVFS), але без того, щоб додатки в робочому навантаженні не пропустили свої дедлайни (як показано на рисунку 2).

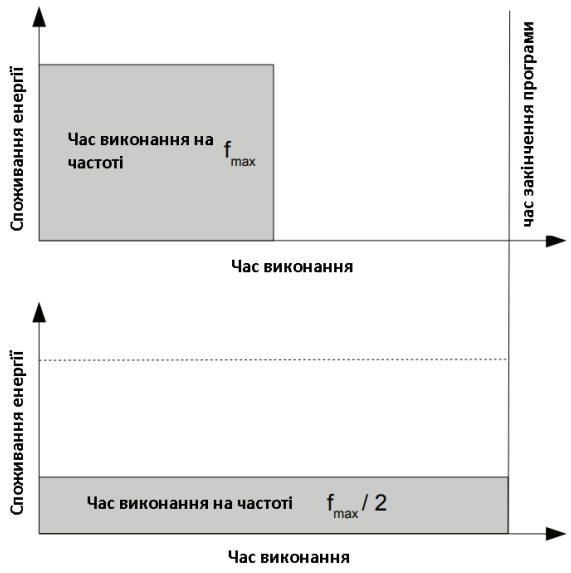


Рисунок 1.2- Виконання завдання на різних напругах

На цьому рисунку показано два завдання, що працюють на різних рівнях напруги та частоти (DVFS).

На верхньому графіку завдання виконується на повній швидкості і завершується задовго до дедлайну. На нижньому графіку частота процесора (і напруга живлення) знижена вдвічі, щоб завдання встигло завершитися до дедлайну, що призводить до меншого споживання енергії.

Цей підхід ґрунтується на спостереженні, що завершення програми до дедлайну, а потім простоювання (наприклад, через DPM), є менш енергоефективним, ніж запуск завдання на нижчій тактовій частоті і завершення його вчасно.

Автори в роботі [11] пропонують оптимальний статичний (автономний) алгоритм планування для залежних завдань. Інший статичний алгоритм планування представлений в [12], який спирається на той факт, що середня активність комутатора завдання (коефіцієнт Cl -Nsw в рівнянні 1) відома до початку виконання.

Використання цієї додаткової інформації дозволяє запропонованому алгоритму зменшити споживання енергії на 70%. Автори в своїй роботі [13] показують, що використання виключно WCET для гарантування часових обмежень робочого навантаження не дає можливості скористатися перевагами невикористаного обчислювального часу, оскільки фактичний і найгірший час виконання в додатках реального часу може суттєво відрізнятися.

Їх експериментальна оцінка демонструє, що запропоновані динамічні та спекулятивні рішення, які використовують невикористаний час обчислень, досягають до 50% зниження енергії порівняно зі статичним алгоритмом (на основі WCET).

Два алгоритми планування, запропоновані в [14] (призначені для незалежних та залежних задач відповідно), спрямовані на подолання основного обмеження раніше запропонованих методів [11], [12], [13].

Як і пропозиції в роботі [13], алгоритми планування, представлені в [14], також використовують невикористаний час обчислень для динамічного регулювання швидкості процесора, але здатні розподіляти цей часовий проміжок між процесорами.

Ще одна робота, що використовує DVFS для динамічного керування живленням, представлена в [15]. На відміну від інших робіт у цій галузі, автори не враховували різні ефекти масштабування частоти для різних додатків.

Замість цього вони використовують статичну оцінку енергоспоживання та продуктивності для кожного режиму роботи процесора. Хоча ця проста політика забезпечила значну економію електроенергії, подальші дослідження намагалися врахувати різні характеристики додатків при розробці політик керування енергоспоживанням.

Більшість прогресивних алгоритмів ґрунтуються на спостереженні, що різні програми по-різному реагують на зміну рівня DVFS, причому деякі з них зазнають більших втрат продуктивності, ніж інші.

Зокрема, в роботах [16], а також багато інших [17], [18], [19] показується, що ступінь погіршення продуктивності програми при зниженні частоти мікропроцесорної системи модуля IoT залежить від того, наскільки програма прив'язана до обчислень.

Задачі, які повністю пов'язані з обчисленнями, так звані burn loop [16], [20], показують лінійну залежність між частотою та часом виконання.

Завдання, які повністю пов'язані з пам'яттю, так звані mem [16], [20], демонструють незначне сповільнення при зниженні частоти.

Нарешті, додатки, що знаходяться посередині між цими двома екстремумами, так звані combo [16], [20], мають середню поведінку. Завдання, пов'язані з пам'яттю, менше страждають від зниження частоти процесора, оскільки частота пам'яті не знижується. Вибір правильного налаштування DVFS для комбінованих програм - це тонкий баланс між зменшенням енергоспоживання та збільшенням часу виконання.

В роботі [16] вирішення цієї проблеми пропонують наступним чином - характеризуючи додатки за ступенем їх обмеженості пам'яттю. Для цього оцінюється стек циклів на інструкцію (Cycles Per Instruction, CPI) кожного потоку в режимі онлайн, використовуючи апаратні лічильники продуктивності, доступні в сучасних контролерах та процесорах, які є частинами IoT модулів.

Стек CPI - це розбивка циклів, витрачених на виконання інструкцій, порівняно із зупинками процесора на промахи кешу, промахи TLB та звернення до пам'яті.

Потім кожен потік характеризується метрикою μ, яка є відношенням циклів, які він витрачає на виконання інструкцій, до загальної кількості циклів. Менший μ вказує на обмеженість пам'яті, а більший μ (що наближається до 1) вказує на обмеженість процесора.

1.3 Планування з урахуванням температури

Комп'ютерні мікросхеми, як і будь-який інший вироблений товар, мають температурний діапазон, в якому вони розраховані на роботу. Коли цей діапазон перевищується, матеріали піддаються напруженню за межею толерантності і можуть розплавитися, зламатися, згоріти або іншим чином припинити роботу, що призведе до часткового або повного виходу з ладу мікросхеми з ладу.

Такі ситуації називаються катастрофічними відмовами, і сучасні мікропроцесорні системи мають вбудовані технології, такі як Multi Point Thermal Control або Adaptive Thermal Monitor, щоб уникнути таких випадків.

Ці методи уникнення збоїв можна широко описати, як реактивне терморегулювання. Процесор будь якого модуля, здебільшого спроектований таким чином, щоб справлятися з тепловиділенням "типового робочого навантаження"; однак, деякі робочі навантаження можуть більш агресивно використовувати ресурси, генеруючи більше тепла, ніж може бути розсіяно, що призводить до небезпечного перегріву.

Такі ситуації виявляються за допомогою температурних датчиків, стратегічно розташованих всередині мікросхеми, і коли температура перевищує встановлений виробником поріг, активуються технології терморегулювання. Технології термоменеджменту можуть знижувати частоту і напругу процесора, застосовувати тактовий генератор (робочі цикли) або взагалі зупиняти роботу. Це робиться до тих пір, поки температура не впаде нижче допустимого порогу, після чого

відновлюється нормальна робота. Оскільки ці методи активуються лише після досягнення критичної температури і не роблять нічого, щоб запобігти виникненню таких ситуацій, вони класифікуються як реактивні методи.

Природа реактивних методів керування температурою, які знижують температуру за рахунок зменшення можливостей процесора, може мати значний негативний вплив на продуктивність потоків, якщо вони активні протягом тривалого часу. Було проведено багато досліджень щодо проактивного термоменеджменту, який вживає превентивних заходів, що самі по собі мають мінімальний вплив на продуктивність або взагалі не впливають на неї, гарантуючи, що критичні температури, які могли б викликати дорогі апаратні реакції, ніколи не будуть досягнуті.

1.3.1 Проблема планування з урахуванням температури

Проблему планування з урахуванням теплових характеристик на багатоядерних процесорах які входять в склад IoT модулів або кінцевих пристроїв можна просто визначити наступним чином: Маючи гетерогенний або гомогенний багатоядерний процесор, створити оптимальний розклад, що визначає розподіл завдань між ядрами з контрольними значеннями теплових характеристик, наявних у сучасних багатоядерних процесорів.

Для наведеної вище задачі багатоцільової оптимізації (БЦО) не існує однозначного рішення [21].

На рисунку 1.3 показано приклад задачі МОО. Ця задача має цільові функції, що суперечать одна одній.

Для певної програми та багатоядерного процесора мікропроцесорної системи оптимальний за Парето граничний розв'язок показано кривою AB. Неможливо далі мінімізувати температуру, T, або довжину розкладу, SL, не збільшуючи одну з цих величин.

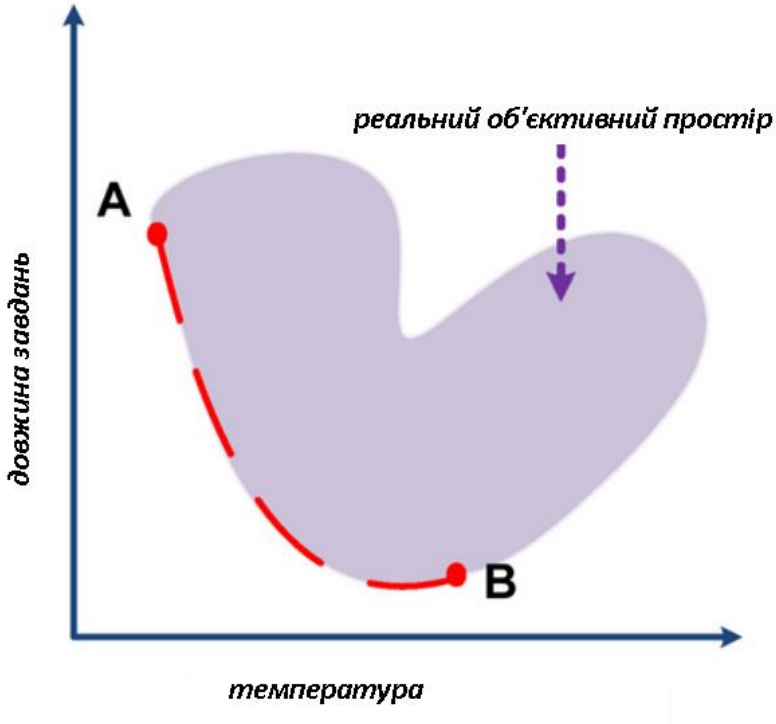


Рисунок 1.3 - Ілюстрація проблеми MOO-NLP

Відповідну робочу точку вздовж кривої AB можна вибрати відповідно до поточних умов, сформулювавши задачу MOO. Практичне значення мають три широкі цілі:

1. Мінімізація довжини розкладу при максимальному температурному обмеженні, відома тут як оптимізація продуктивності з температурним обмеженням (POTC). Це передбачає мінімізацію часу виконання (погіршення продуктивності) для завершення завдань в умовах обмеження заданої розробником температурної стелі.

2 Мінімізувати максимальну або середню температуру контролера або процесора при заданому дедлайні виконання завдання. Це можна розглядати, як протилежний випадок першої потенційної мети, оскільки температура мінімізується до заданого дедлайну. Мінімізація температури потенційно може призвести до прямої або непрямої економії енергії, такий підхід називається оптимізацією температури з обмеженням продуктивності (TOPC).

3. Оптимізувати, як тривалість розкладу, так і тепловіддачу, знаходячи компроміс між цими двома цілями, відомий, як оптимізація продуктивності та оптимізація температури (POTO).

1.3.2 Моделювання задач та їх варіації

При дослідженні планування з урахуванням температури для багатоядерних процесорів вивчаються різні моделі задач. Вони можуть бути широко розбиті на наступні виміри:

1. Незалежні та залежні: Більшість з них включають незалежні завдання, де немає відносин пріоритету між завданнями [22,23, 24,25,26, 27], є найбільш корисними для додатків загального призначення, таких як сервери і системи розумних будинків. Багато наукових, мультимедійних та біоінформатичних додатків складаються з низки пов'язаних між собою задач, з обмеженнями або без них. Для задач з відношеннями пріоритету, вони можуть бути представлені за допомогою DAG (Directed Acyclic Graph - спрямований ациклічний граф).

DAG представляє робочий процес між завданнями програми. У DAG вузол представляє задачу, а ребро від вузла i до вузла j відображає відношення пріоритету від задачі i до задачі j. На рисунку 1.4 (а) зображено приклад DAG з вагами на вузлах і ребрах, що описують обчислювальні та комунікаційні витрати відповідно. Пріоритет завдань в DAG повинен зберігатися, і це може додати рівень складності до проблеми, яка вивчається в [22,28]. На рисунку 1.4 (б) показано потік виконання вузлів DAG на 3-ядерній машині, коли завдання планується на основі залежностей в DAG.

2. Реальний час проти нереального часу: У багатьох середовищах завдання мають обмеження на прибуття та/або дедлайн [5,42,43]. Якщо існують обмеження щодо термінів, то це можуть бути системи жорсткого реального часу або системи м'якого реального часу. Жорсткі системи реального часу вимагають дуже суворого дедлайну, що означає, що завершення завдання після його дедлайну вважається марним. Для порівняння, дедлайн в м'яких системах реального часу не є строго

обов'язковим. Допускається невелика кількість затримок, але це призведе до деякої деградації продуктивності або зниження якості обслуговування. Приклади завдань жорсткого реального часу включають (1) виявлення критичних станів, (2) контроль критичних компонентів системи та (3) збір сенсорних даних. Приклади завдань реального часу включають (1) відображення повідомлень на екрані, (2) графічні дії і (3) збереження даних звітів [29].

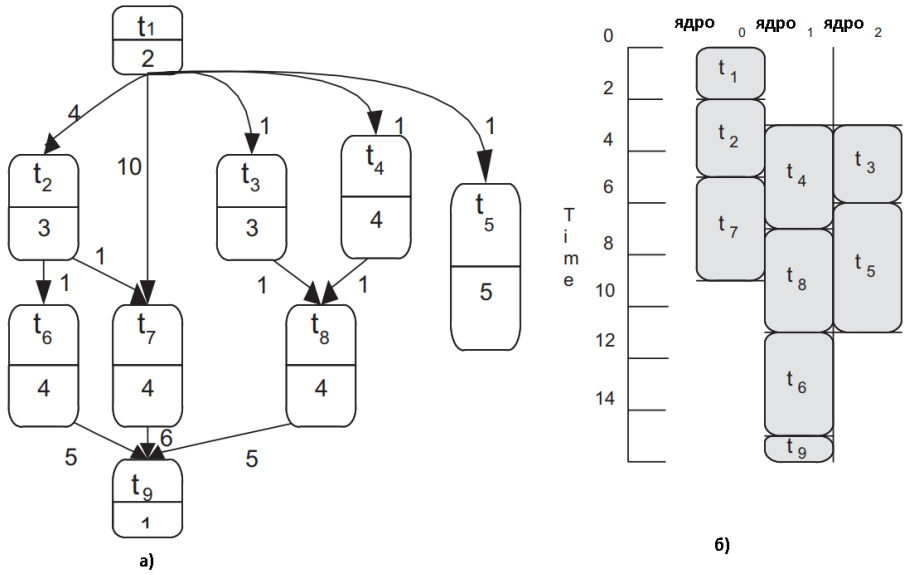


Рисунок 1.4 - Репрезентативний DAG (а); Потік виконання, що показує завдання DAG на різних ядрах (б)

3. Періодичні завдання, аперіодичні завдання та періодичні завдання: Періодичні завдання включають нескінченну послідовність однакових екземплярів. Екземпляри періодичних завдань відбуваються через рівні проміжки часу. Час активації першого екземпляра - це фаза, позначається через ϕ. Для періодичної задачі з періодом T час активації k-го екземпляра задається через $\phi + (k - 1)T$. На рисунку 1.5 . 3(а) показано періодичну задачу, де WCET - найгірший час виконання, а D - відносний дедлайн. Відносний дедлайн у багатьох випадках визначається як кінець періоду.

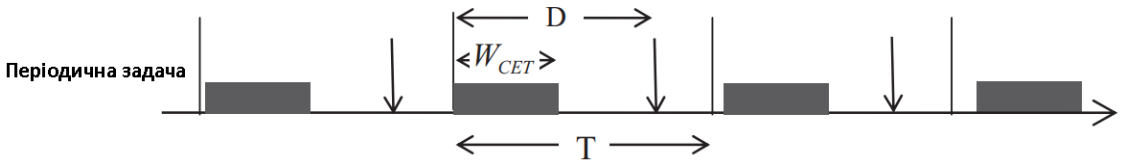


Рисунок 1.5 – Періодична задача

Аперіодичні завдання включають нескінченну послідовність робіт, які активуються нерегулярно. На рисунку 1.6 показано жорстку аперіодичну задачу з часом надходження, дедлайном і WCET.

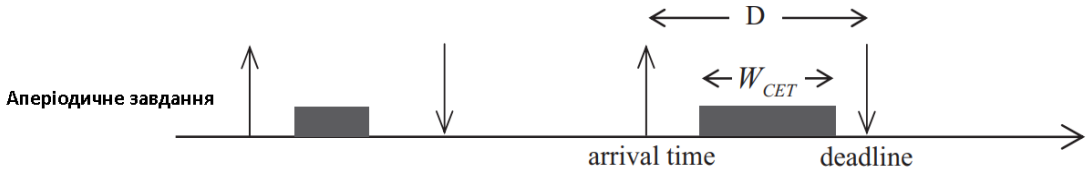


Рисунок 1.6 – Аперіодичне завдання

Дедлайн м'якої аперіодичної задачі не є строго обов'язковим. Епізодичні завдання, подібно до аперіодичних, включають нескінченну послідовність однакових робіт, які активуються нерегулярно. Однак, визначено мінімальний час між двома послідовними активаціями. На рисунку 1.7 показано спорадичну задачу з її відносним дедлайном, найгіршим часом виконання (WCET) та мінімальним часом між надходженнями .

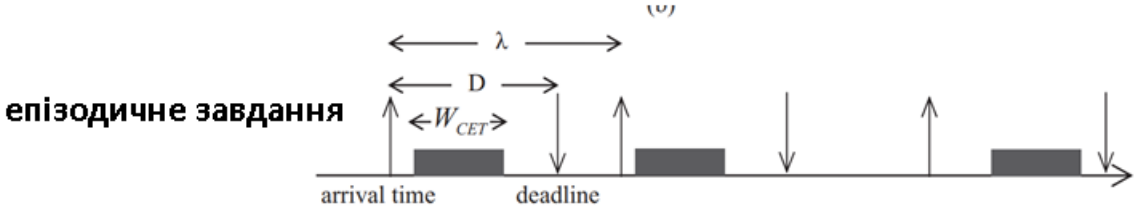


Рисунок 1.7 – Епізодичне завдання

Якщо різні параметри навантаження, такі як порядок виконання, залежності між завданнями та дедлайн, пов'язаний з кожним завданням, відомі апріорі, то

зазвичай розглядається статичне планування, як в роботах [22,28]. Однак, коли ці параметри невідомі, необхідно розглядати випадок динамічного або онлайн планування [23,30, 32].

1.4 Багатоядерні системи та теплове моделювання

1.4.1 Багатоядерні процесори

Більшість сучасних процесорів мають величезну кількість транзисторів на одному кристалі. Незважаючи на те, що багато архітектур підтримують спільний доступ до пам'яті або кешу, одночасний доступ до одного конкретного блоку пам'яті все ще обмежений одним ядром.

Ієрархія пам'яті є важливою на високих частотах. Локальність доступу до даних необхідна для досягнення хорошої продуктивності на процесор в модулі IoT. Завдяки оптимальному використанню функціональних блоків, черг попередньої вибірки даних і черг команд, такі процесори або контролери можуть скоротити час циклу команд до декількох наносекунд. У зв'язку зі збільшенням енергоспоживання та зростанням попиту на обчислювальну потужність, багато виробників апаратного забезпечення зосереджують увагу на системах управління температурою своїх продуктів за допомогою різних методів.

1.4.2 Визначення температури

Керування тепловим режимом вимагає ефективного визначення теплового профілю обчислювальної системи під час роботи. Методи вимірювання температури процесора в модулі IoT можна розділити на три категорії:

1. Методи на основі датчиків: Ці методи використовують теплові датчики на кристалі для отримання температури в реальному часі [32].

2. Методи на основі теплових моделей: Ці методи використовують теплові моделі [34,35, 30] для отримання температури. Методи на основі моделей

забезпечують моделювання на основі електрично-теплового дуалізму за допомогою використання кускової RC-моделі.

3. Методи на основі лічильників продуктивності: Ці методи оцінюють температуру системи, використовуючи значення, зчитані з певних регістрів [44].

Основною перевагою методів на основі датчиків є низькі обчислювальні витрати. Сенсорні методи також більш придатні для визначення температури в реальному часі для підтримки механізмів DTM. Недоліком сенсорного методу є неточність і шум від вихідних даних, що генеруються датчиком температури.

Методи на основі моделей, навпаки, можуть точно оцінити температуру. Запропоновано широкий спектр температурних моделей, які можна умовно поділити на низькорівневі та високорівневі.

1.4.3 Міграція потоків

Міграція потоків є поширеною технікою, яка використовується для вибіркового запуску, витіснення або перенесення запланованого або виконуваного потоку на інше ядро на основі його теплового або енергетичного профілю.

Накладні витрати виникають через витрати на зв'язок та інші фактори, такі як оновлення в адресному просторі при передачі стану між ядрами. Ці накладні витрати необхідно враховувати при аналізі компромісу між вигодами та втратами в продуктивності, що виникають при міграції потоків. У роботі [30] розглядаються деякі питання призначення та міграції потоків. Далі досліджується, як ці проблеми можна вирішити за допомогою операційної системи на мікросхемі з одночасно-багатопоточними ядрами (SMT). У роботі [23] пояснюються питання міграції потоків, такі як частота міграції та роль штрафів холодного кешу, які можуть впливати на продуктивність. В роботі [23] розглянуто п'ять схем міграції потоків:

1. Схема ротації мігрує потік з ядра i на ядро (i + 1) mod N, де N - кількість ядер процесора.

2. Міграція потоків на основі температури використовує заміну потоків, таким чином, що потік, який виконується на найгарячішому ядрі, обмінюється з потоком,

який виконується на найхолоднішому ядрі. Аналогічно, потоки на інших гарячих ядрах обмінюються з потоками на аналогічних за рангом (друге найгарячіше, друге найхолодніше) холодних ядрах відповідно до їхніх температур.

3. Міграція потоків на основі лічильника ранжує ядра на основі температурних дельт (різниця між найвищою гарячою точкою і другою найвищою гарячою точкою). Цей список потім ітерується зверху вниз, щоб передбачити найгарячіший потік і перемістити його на найхолодніше ядро.

4. Покращена схема на основі лічильників зменшує непотрібні міграції. Вона впорядковує ядра у списку за температурою, але не виконує міграцію, якщо температура нижча за граничну більш ніж на 1°C.

5. Схема на основі потужності впорядковує ядра у списку за зростаючою температурою, а потім впорядковує потоки на основі їх потужності. Потім вона мігрує потік з найбільшим енергоспоживанням на ядро з найнижчою температурою, але не виконує міграцію, якщо температура нижча за граничну більш ніж на 1°C.

1.4.4 Оптимізація продуктивності з урахуванням теплових обмежень (POTC)

Продуктивність модуля IoT в складі якого є багатоядерний CPU можна виміряти різними способами, але найпоширенішими параметрами, які використовуються для оптимізації, є час виконання заданого набору задач і пропускна здатність системи.

Як пояснювалося вище, набори задач, що виконуються багатопроцесорними системами, можуть бути незалежними, залежними, в м'якому або жорсткому реальному часі. Задача POTC полягає в оптимізації метрики продуктивності при заданих теплових обмеженнях, таких як максимально допустима температура ядра, середня температура всіх ядер або якась інша функція, пов'язана з температурою ядер. Деякі з останніх досліджень, які фокусуються на певній формі POTC, можна знайти в [45].

1.5 Незалежні та залежні задачі на багатоядерних IoT пристроях

Припустимо, що "Tk" являє собою набір незалежних завдань Tk = (Tk1, Tk2, Tk3, Tk4, ..., Tkn). Кількість команд для кожної задачі відома апріорі і позначається C = (c1, c2, c3, c4, ..., cn).

Нехай uj - величина, обернена до j-го рівня напруги, а xij - змінна рішення, значення якої дорівнює "1", якщо i-та задача буде виконуватися з використанням j-го рівня напруги. Система має p ядер і ˇ - середня кількість інструкцій за такт, а - константа, яка пов'язує значення рівня напруги з тривалістю такту. k - температура k-го ядра, а th і o - обмеження на максимальну і середню температуру системи. ˛p,k - коефіцієнт, який пов'язує температуру p-го ядра з рівнем потужності k-го ядра, а Pk - потужність, що споживається k-м ядром. Тоді, для сценарію POTC напруги повинні бути призначені для кожної задачі так, щоб мінімізувати загальний час виконання (1) та/або максимізувати пропускну здатність (2), тобто

$$Min \sum_{i=1}^{n} \sum_{j=1}^{L} \gamma \cdot x_{ij} \cdot c_i \cdot u_j \quad (1)$$

або

$$Max \sum_{i=1}^{n} \beta \cdot f_i \quad (2)$$

1.5.1 Залежні задачі

Якщо задача, яку мають виконувати багатоядерні процесори, складається із залежних задач, то оптимізація продуктивності повинна проводитись при додаткових обмеженнях. Припустимо, що набір задач представлений графом задач, а зв'язок між задачами записаний у вигляді булевої змінної ei,j = 1, якщо задача j може бути виконана тільки після завершення задачі i. Оптимізація продуктивності в цій задачі знову ж таки може полягати у зменшенні часу виконання, як і в (3), не впливаючи на залежність між завданнями. Звідси випливає додаткове обмеження,

що час початку j-го завдання має бути більшим за час завершення і-го завдання. Це призведе до того, що

$$Start\text{-}time_j > end\_time_i, \text{ if } e_{ij}=1 \; \forall i,j \tag{3}$$

1.6 Постановка задачі

Із збільшенням обчислювальної потужності та функціональних можливостей пристроїв IoT зростає і втрата тепла. Це створює серйозні проблеми для підтримки оптимальних температурних режимів пристроїв. Підвищення температур в IoT пристроях може призвести до скорочення терміну їхньої роботи, збоїв та погіршення надійності. Оптимізація теплових режимів стає ключовим аспектом для забезпечення тривалої та надійної роботи. Також високі температури призводять до збільшення споживання енергії. Зменшення пікових температур через оптимізацію завдань може призвести до зменшення енергоспоживання та сприяти більш екологічному використанню ресурсів.

Тому, розробка методів, які зменшать пікові температури в IoT пристроях через міграцію та заміну завдань, є актуальною та ключовою задачею для подальшого розвитку та стабільності Інтернету речей у різноманітних сферах таких як індустрія 4.0.

Метою дипломної роботи є розробка методу зниження пікових температур в IoT пристроях за допомогою міграції та заміни завдань

Досягнення цієї мети зумовило потребу теоретичних розробок, визначення та послідовного вирішення таких завдань:

1. дослідити методи планування завдань для IoT пристроїв;
2. дослідити типи базові принципи розсіювання потужності мікропроцесорних та мікроконтролерних системах;
3. дослідити методи планування з урахуванням температури;
4. дослідити методи теплового моделювання в багатоядерних системах;
5. дослідити незалежні та залежні задачі на багатоядерних IoT пристроях
6. розробити метод зниження пікових температур в іот пристроях за допомогою міграції та заміни завдань;
7. розробити алгоритм планування з урахуванням теплового режиму на основі динамічної теплової моделі;

8. розробити та реалізувати на основі методу зниження пікових температур в IoT пристроях за допомогою міграції та заміни завдань розробити модель оцінки методів управління ресурсами та провести оцінку виконання моделі;

Завдання 1-4 розглянуто в параграфах 1.1, 1.2, 1.3, 1.4, 1.5 відповідно, а виконання завдань 6-8 представлені у параграфах 2.1-3.3.

Висновки до розділу 1:

Як показано в попередніх розділах та в огляді робіт, необхідність, переваги та можливість зменшення тепловиділення процесора який є в складі IoT пристрою є актуальною задачею.

З метою зменшення енергоспоживання процесора було проведено багато досліджень. Цікавим майбутнім напрямком досліджень в області планування з урахуванням теплового режиму буде зв'язок зменшення тепловиділення з економією енергії і, в кінцевому підсумку, економією загальної вартості експлуатації систем.

Розглянуті роботи включають статичне і динамічне планування для багатоядерних процесорів, включаючи системи і MPSoC, а також балансування навантаження з урахуванням температури і планування в реальному часі на багатоядерних системах.

В даному розділі було проаналізовано базові принципи розсіювання потужності мікропроцесорних та мікроконтролерних системах та проблеми з якими стикаються розробники. Також досліджено принципи керування енергоспоживанням під час виконання завдань, динамічне масштабування напруги/частоти та динамічне керування живленням, та як поводять себе процеси при маніпуляцією живлення пристрою. Крім того розглянуто методи планування завдань з урахуванням температури та проблеми з якими стикаються розробники та виробники мікроконтролерів та процесорів для IoT модулів. Також проведено аналіз медодів попереднього моделювання завдань для енергоефективності та методи теплового моделювання для багатоядерних сситем в складі IoT модулів.

2 МЕТОД ЗНИЖЕННЯ ПІКОВИХ ТЕМПЕРАТУР В ІОТ ПРИСТРОЯХ ЗА ДОПОМОГОЮ МІГРАЦІЇ ТА ЗАМІНИ ЗАВДАНЬ

2.1 Фізично-інформоване теплове моделювання за допомогою декомпозиції

POD - це фізично-інформована модель, яка здатна представляти складну теплову задачу в часі і просторі, використовуючи навчені режими POD разом з рівнянням теплопровідності [38, 39].

Режими витягуються з даних теплового розв'язку фізичної області в процесі навчання. Дані, що використовуються для навчання, повинні включати ряд параметричних варіацій, щоб інформувати модель про реакцію фізичної величини в області в різних умовах. Після навчання створені режими налаштовуються на параметричні варіації, такі як граничні умови (ГУ) і варіації потужності. Далі GP надає фізично обґрунтовані вказівки для покращення точності та ефективності моделі

2.1.1 Побудова режимів POD

Режими POD оптимізуються шляхом максимізації середньоквадратичного внутрішнього добутку даних теплового розв'язку на режими по всій області з урахуванням динамічних або статичних параметричних варіацій граничних умов та внутрішніх джерел енергії. У дослідженні теплові дані збираються на кожному часовому кроці моделювання в MCE DNS для збору даних.

Цей процес оптимізації призводить до проблеми власних значень, яка описується рівнянням Фредгольма, показаним у рівнянні (4):

$$\int_{\Omega} R(\underset{r}{\rightarrow},\underset{r}{\rightarrow})\, \varphi\, (\underset{r}{\rightarrow}) d \underset{r}{\rightarrow} = \lambda \varphi(\underset{r}{\rightarrow}), \qquad (4)$$

де λ - власне значення, що відповідає режиму POD (тобто власна функція φ), а $R(\underset{r}{\rightarrow},\underset{r}{\rightarrow})$ - двоточковий кореляційний тензор. Цей двоточковий кореляційний тензор показано в рівнянні (5):

$$R(\underset{r}{\rightarrow},\underset{r}{\rightarrow}) = \langle T(\underset{r}{\rightarrow} t) \otimes T(\underset{r}{\rightarrow} t) \rangle \quad (5)$$

де $\otimes$ - тензорний оператор, а $\langle\rangle$ - середнє значення за кількістю наборів теплових даних. З режимами, отриманими в процесі навчання, включаючи збір теплових даних і розв'язання задачі на власні значення, як показано в рівнянні (4), температура може бути представлена рівнянням (6):

$$T(\underset{r}{\rightarrow},t) = \sum_{i=1}^{M} a_i(t), \varphi_i(\underset{r}{\rightarrow}) \quad , \quad (6)$$

де M - кількість режимів POD, обраних для реконструкції температурного поля.

2.1.2 Проекція теплової задачі

GP використовується для побудови моделі POD шляхом проекції рівняння теплопередачі на простір POD, як показано в рівнянні (7):

$$\int_{\Omega} \left(\varphi_i \left(\underset{r}{\rightarrow}\right) \frac{\partial_p CT}{\partial t}\right) + \nabla\varphi_i \left(\underset{r}{\rightarrow}\right) \cdot k\nabla T) d\Omega = \int_{\Omega} \varphi_i \left(\underset{r}{\rightarrow}\right) P_j \left(\underset{r}{\rightarrow}, t\right) d\Omega -$$
$$\int_{\Omega} \varphi_i \left(\underset{r}{\rightarrow}\right)(-k\nabla T \cdot \underset{n}{\rightarrow}) dS, \quad (7)$$

де k, ρ та C - теплопровідність, густина та питома теплоємність відповідно. Pd($\vec{r}$, t) - внутрішня густина потужності, S - гранична поверхня, а $\vec{n}$ - вектор зовнішньої нормалі до граничної поверхні.

З вибраними режимами POD рівняння (4) можна переписати у вигляді M-вимірного звичайного диференціального рівняння для ai(t), як показано в рівнянні (8):

$$\sum_{i=1}^{M} C_{i,j} \frac{da_i(t)}{dt} + \sum_{i=1}^{M} g_{i,j} a_i(t) = P_j, j = 1 \; to \; M \quad (8)$$

де ci,j та gi,j - елементи матриць теплоємності та теплопровідності в просторі POD, які визначаються в рівняннях (9) та (10) відповідно як:

$$c_{i,j} = \int_{\Omega} \rho C \varphi_i(\underset{r}{\rightarrow})\, \varphi_j(\underset{r}{\rightarrow}) d\Omega \tag{9}$$

$$g_{i,j} = \int_{\Omega} k \nabla \varphi_i(\underset{r}{\rightarrow}) \cdot \nabla\, \varphi_j(\underset{r}{\rightarrow}) d\Omega \tag{10}$$

Pj у рівнянні (7) є потужністю джерела живлення для j-го режиму POD у просторі POD і описується нижче в рівнянні (11):

$$P_j = \int_{\Omega} \varphi_i(\underset{r}{\rightarrow}) P_d\left(\underset{r}{\rightarrow}, t\right) d\Omega - \cdot \int_{\Omega} \varphi_i(\underset{r}{\rightarrow})(-K \nabla \mathrm{T} \cdot \underset{n}{\rightarrow}) dS \cdot \tag{11}$$

Після того, як отримано споживану потужність, можна попередньо оцінити потужність внутрішнього джерела енергії в просторі POD, наведену в рівнянні (8). Для граничного джерела тепла в рівнянні (8) дна підкладки моделюється конвекційним теплообміном з постійним коефіцієнтом тепловіддачі і температурою навколишнього середовища 45 °C (Tamb). Всі інші границі є адіабатичними. Коефіцієнти в рівнянні (6) також можуть бути попередньо оцінені після визначення режимів. Розв'язавши ai(t) з рівняння (5) в симуляції POD, температурний розв'язок можна передбачити з рівняння (3).

2.2 Алгоритм планування з урахуванням теплового режиму на основі динамічної теплової моделі POD

Даний алгоритм використовує пару температурних порогів для обмеження максимальної температури ядра. Після вибору та призначення завдань, використовується для прогнозування температури всіх ядер процесора в IoT модулі, щоб ядра могли простоювати, коли температура в них перевищує поріг. Як тільки прогнозована температура досягає порогового значення, ядра, температура яких перевищує гарячий температурний поріг, TH, переходять у режим очікування.

Простоювання триває до тих пір, поки ядро не досягне "холодного" порогу, TC, після чого йому дозволяється відновити виконання. Пропонований алгоритм, який представляє метод спирається на динамічну теплову модель для прогнозування температури процесора з часом.

З іншого боку, стаціонарна модель прогнозує лише температуру, при якій процесор досягне рівноваги, враховуючи певну потужність розсіювання. На рисунку 3.1 показано прогнози динамічної теплової моделі поряд з прогнозами стаціонарної теплової моделі.

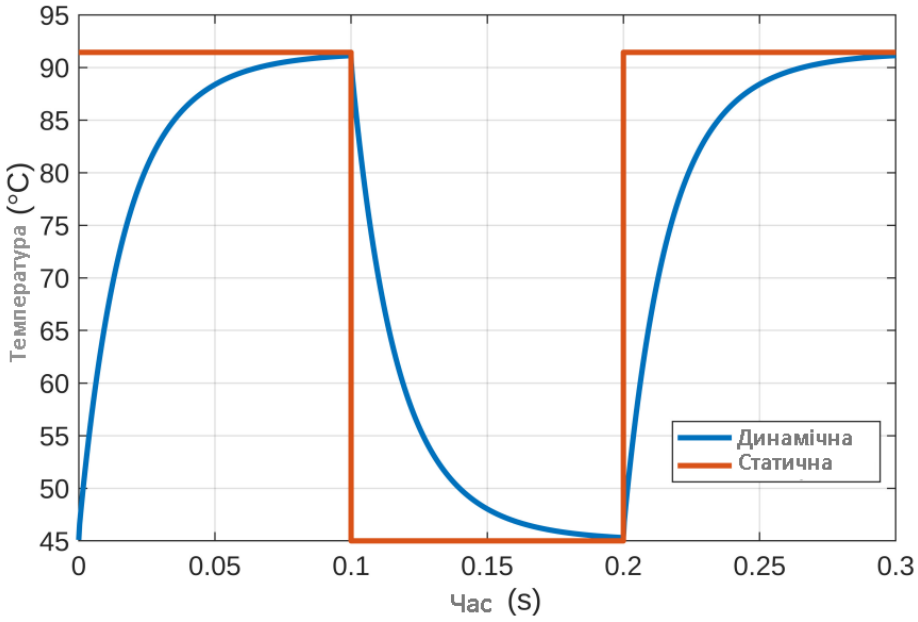


Рисунок 3.1 - Динамічне прогнозування температури в порівнянні зі стаціонарним станом

Коли потужність розсіювання висока, як показано при t = 0s і t = 0.2s, стаціонарна модель прогнозує температуру 91.43°C. Однак, як показують динамічні прогнози, ядру потрібно значний час, щоб наблизитися до цієї температури. Враховуючи цю різницю, планувальник може приймати неправильні рішення на основі стаціонарних прогнозів, якщо тільки швидкість прийняття рішень планувальником не буде надзвичайно повільною.

Однак, як час виконання більшості завдань (WCET) набагато менший, ніж час, необхідний процесору для наближення до стаціонарної температури. Таким чином, динамічне прогнозування температури необхідне для того, щоб планувальник приймав рішення на основі точних прогнозів теплового стану процесора.

Алгоритми 1, 2, 3 і 4 представляють основну концепцію методу. На основі цих концепцій було реалізовано систему планування, яка використовує новий алгоритм.

У запропоновано алгоритмі (рисунок 2.1) позначення π представляє процесорну платформу, де π = {π1, π2, ..., πm} - множина з m ядер процесора.

```
1:  function Plan_temp (τ, T_C, T_H, t_end)
2:      t_curr ← 0; Λ_prev ← ∅; σ ← [];
3:      while t_curr < t_end do
4:          τ' ← SELECT_TASKS(τ, π, Ψ)
5:          Λ ← ASSIGN_TASKS(τ', π, Λ_prev, Ψ)
6:          Λ_prev ← Λ
7:          append(σ, (t_curr, Λ))
8:          do
9:              t_curr ← t_curr + Δt
10:             T_π ← PREDICT_TEMP(P_σ, t_curr)
11:             Ψ ← UPDATE_STATES(π, Ψ, T_π, Λ)
12:         while max(T_πi(t_curr), ∀π_i ∈ π) < T_H and t_curr < t_end
13:     return σ
```

Рисунок 2.1 – Псевдо код алгоритму планування з урахуванням теплового режиму

Температури в момент часу t - Tπi (t), πi ∈ π, де1 ≤ i ≤ m. Множину задач позначено через τ , так що τk ∈ τ позначає k-ту задачу. Відображення або розподіл задач з τ на ядра процесора в π позначається через Λ. Λπi представляє задачу, яка призначена ядру процесора i, де πi належить відсортованій підмножині π, яка має допустимі теплові стани.

Позначення Ψ у алгоритмі представляє множину станів ядер процесора, як показано на рисунку 2.2

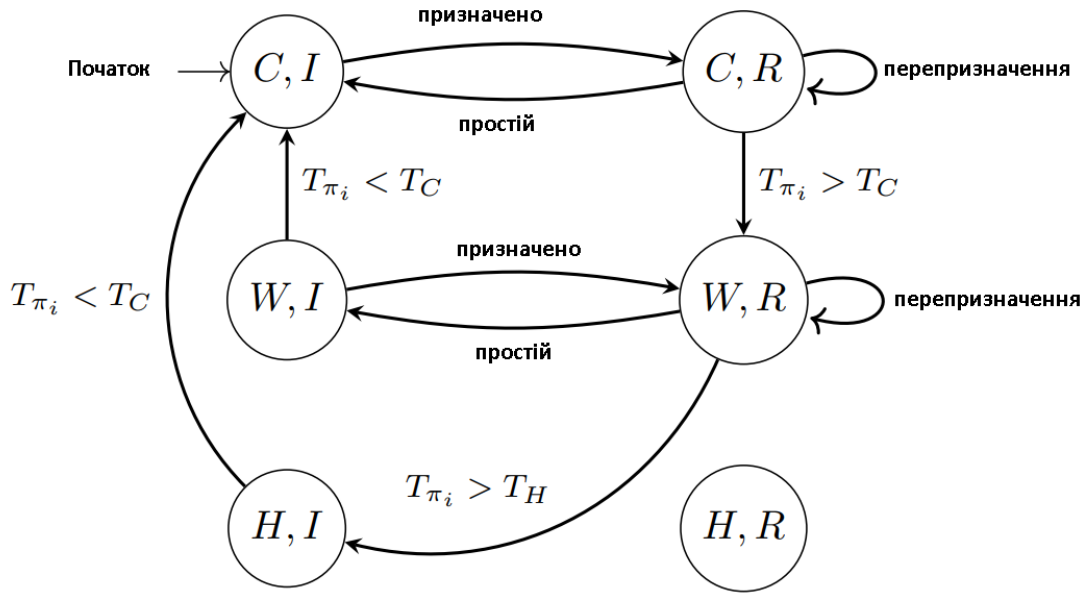


Рисунок 2.2 - Діаграма переходів станів для ядер процесора

Ψπі використовується для позначення стану і-го ядра процесора, πі .

Для позначення теплового стану ядер процесора описано набір станів та переходів між ними, щоб керувати поведінкою процесора. На рисунку 3.2 показано можливі стани процесора Ψ та переходи між ними. Кожен стан складається з пари літер, з яких перша літера позначає тепловий стан ядра, а друга - стан виконання ядра.

Можливі теплові стани C, W та H, які відповідають холодному, теплому та гарячому відповідно. Можливі стани виконання - I та R, де I позначає непрацююче ядро, а R - працююче ядро. Визначення станів кожного ядра процесора і переходів між ними визначає, як алгоритм повинен поводитися з ядрами процесора в залежності від їх стану і які дії повинні бути виконані в залежності від їх стану. Таким чином, можна уникнути призначення завдань ядрам, які перебувають у невідповідному стані.

Основна концепція призначення ядер в залежності від їх теплового стану представлена в Алгоритмі 2 (рисунок 3.3).

```
1: function UPDATE_STATES(π,Ψ,T_π,Λ)
2:     for π_i ∈ π do
3:         run_state ← R
4:         if Λ_πi = ∅ then
5:             run_state ← I
6:         if T_πi < T_C then
7:             Ψ_πi ← (C, run_state)
8:         else if T_C < T_πi < T_H then
9:             Ψ_πi ← (W, run_state)
10:        else
11:            Ψ_πi ← (H, I)
12:    return {Ψ_πi where π_i ∈ π}
```

Рисунок 2.3 - Псевдокод алгоритму управління станом ядра з урахуванням теплового режиму

Позначення σ використовується для опису впорядкованої множини призначень між завданнями та ядром у поєднанні з часом їх виконання (розкладом). Крім того, для позначення потужності, що розсіюється за певним розкладом, використовується позначення Pσ. Потужність, що розсіюється ядрами процесора, залежить від призначення задачі, а потужність, що розсіюється кожною задачею в даний момент часу, залежить від того, скільки часу виконання задачі було надано розкладом σ. Таким чином, Pσ містить потужність, що розсіюється процесором при виконанні розкладу σ з плином часу.

Алгоритм 2 описує підпрограму UPDATE STATES, яка включає в себе процедуру управління станами ядер і викликається в Алгоритмі 1. Як показано на рисунку 2.3, всі ядра починають роботу в холодному, неактивному стані (C, I). З цього стану, коли ядру призначається завдання, воно переходить у стан прохолодної роботи, (C, R).

Зі стану (C, R) ядро може повернутися в (C, I), якщо воно простоює через виконання або перепризначення завдання. Однак, якщо температура ядра Tπi перевищує TC, ядро переходить у стан (W, R), який позначає тепле, працююче ядро.

Теплі стани означають, що ядро перетнуло TC, але ще не перетнуло TH, тобто ядро знаходиться в перехідному стані. Подібно до переходів з (C, R), ядро в стані (W, R) можна простоювати, переводячи його в стан (W, I) (тепле, бездіяльне), але якщо його температура перевищує TH, ядро змушене простоювати і переходить в стан (H, I) (гаряче, бездіяльне).

Ядрам у стані (H, I) заборонено виконувати будь-які завдання до тих пір, поки їх температура не впаде нижче TC і вони не повернуться в стан (C, I). Це обмеження гарантує, що після того, як завдання перевищить поріг високої температури, воно повинно охолонути, перш ніж відновити виконання.

Це гарантує, що жодне ядро не буде швидко чергувати виконання завдання з простоєм. Стан (H, R) (гарячий, працюючий) відокремлений від інших станів і не має переходів до нього, оскільки цей стан заборонений. Як тільки температура ядра перевищує TH, ядро переходить у стан простою.

В алгоритмі 1 формально описано основну концепцію алгоритму планування з урахуванням теплового режиму.

- Алгоритм приймає на вхід:
- τ - набір завдань, які потрібно запустити;
- TC - поріг температури охолодження процесора;
- TH - поріг температури перегріву процесора;
- і tend - час, на який потрібно побудувати розклад.

Функція повертає графік σ, який можна зберегти для використання у системі планування. Існує чотири підпрограми, які викликаються: UPDATE STATES, SELECT TASKS, ASSIGN TASKS і PREDICT TEMP. PREDICT TEMP бере динамічну карту потужності поточного розкладу і час, для якого потрібно спрогнозувати, і прогнозує температуру процесора для кожного ядра процесора в цей час, використовуючи теплову модель POD. Оновлення станів описано в алгоритмі 2.

У алгоритмі планування з урахуванням теплового режиму керування температурою здійснюється в основному за допомогою двох порогових значень, але

вибір завдань з черги і основних завдань може мати значний вплив на створений розклад і на можливість планування набору завдань.

Відбір завдань здійснюється шляхом надання пріоритету виконанню завдань з більшим залишковим часом виконання (T imeLef t(τi)). Це допомагає уникнути невиправданого простою завдань, які не можуть бути виконані до встановленого терміну.

Алгоритм вибору завдань (підпрограма SELECT TASKS) описана в алгоритмі 3 (рисунок 2.4)

1: **function** SELECT_TASKS(τ,π,Ψ)
2: $n\_tasks \leftarrow n(\{\pi_i \in \pi | \Psi_{\pi_i} \neq (H, I)\})$
3: $\tau' \leftarrow sort(\tau | TimeLeft(\tau_i) \geq TimeLeft(\tau_{i+1}))$
4: $\tau_{run} \leftarrow \{\}$
5: **while** $n(\tau_{run}) < n\_tasks$ **do**
6: $\tau_{run} \leftarrow \tau_{run} \cup \{\tau'[n(\tau_{run})]\}$
7: **return** τ_{run}

Рисунок 2.4 - Алгоритм вибору завдань

Основна концепція цього алгоритму полягає в тому, що для кожного ядра процесора, яке не перебуває в стані (H, I), слід вибрати завдання з τ , надаючи пріоритет завданням з більшим залишковим часом виконання.

Алгоритм 4 (підпрограма ASSIGN TASKS) (рисунок 2.5) описує призначення вибраних завдань на доступні ядра.

Цей алгоритм спочатку сортує ядра в порядку зростання температури, щоб більш холодним ядрам були призначені завдання, які вимагають більшого часу виконання. Ця схема пріоритизації має на меті забезпечити максимальний час виконання для завдань, які цього потребують, оскільки алгоритм 3 гарантує, що завдання з найбільшим залишковим часом виконання отримують пріоритет. Взаємодія та ефекти цих двох алгоритмів є потенційним майбутнім напрямком досліджень.

```
1: function ASSIGN_TASKS(τ_run, π, Λ_prev, Ψ)
2:     π' ← sort({π_i ∈ π | Ψ_πi ≠ (H, I)} | T_πi ≤ T_πi+1)
3:     for τ_k ∈ τ_run do
4:         for π_i ∈ π' do
5:             if Λ_πi = ∅ then
6:                 Λ_πi ← τ_k
7:                 break
8:     return {Λ_πi where π_i ∈ π'}
```

Рисунок 2.5 - Алгоритм розподілу завдань між основними завданнями

2.3 Складність виконання алгоритму планування з урахуванням теплового режиму

Алгоритм 1 - це цикл з лінійною часовою складністю на основі tend/Δt, який викликає алгоритми 2, 3 та 4. Алгоритм 2 використовує лінійний цикл через ядра процесора для оновлення станів ядер у Ψ. Алгоритм 3 сортує задачі в не зростаючому порядку на основі часу, що залишився до їх виконання, перш ніж використовувати лінійний цикл, щоб взяти достатню кількість задач для заповнення доступних ядер процесора.

Найбільші витрати в Алгоритмі 3 пов'язані з сортуванням завдань на основі часу, що залишився, яке за допомогою звичайних ефективних алгоритмів сортування може бути досягнуто за час Θ(n log(n)), де n - кількість завдань, які потрібно відсортувати.

Алгоритм 4 сортує ядра процесора на основі їх температури перед парою вкладених циклів, які перебирають вибрані задачі та доступні ядра процесора. Таким чином, Алгоритм 4 працює зі складністю Θ(m log(m) + qr), де m - кількість ядер процесора в π, q - кількість задач в τrun, і r - кількість ядер процесора в π′.

Кількість ядер процесора можна вважати невеликою у загальному випадку планування роботи процесора, тоді як кількість задач не є достатньо великою, щоб мати великий вплив на обчислювальну вартість сортування множини задач. Таким чином, основним джерелом обчислень є Алгоритм 1 з його циклом, який викликає

інші три алгоритми. Отже, можна сказати, що основний алгоритм в цілому має лінійну часову складність, залежну від tend/Δt.

2.3.1 Розподіл завдань за допомогою підходів до пакування контейнерів

Варто зазначити, що і алгоритм 1, і алгоритм 2 припускають, що завдання в реальному часі можуть бути вільно розподілені відповідно до обчислювальних можливостей кожного ядра. Тому результати алгоритму 1 та алгоритму 2 фактично є верхньою межею енергоефективності.

Насправді, завдання реального часу не можуть бути розділені довільно, а відображення завдань реального часу на декілька ядер саме по собі є важкою задачею. Одним з поширених евристичних підходів до розбиття багатоядерних задач є перетворення їх на задачу пакування сміттєвих контейнерів [46].

У підході LALB після визначення оптимальної групи активних ядер та швидкості їх обробки визначається ємність кошика. Проблема полягає в тому, щоб упакувати об'єкти (завдання), кожен з яких має різний розмір (утилізацію), в кошики так, щоб необхідна кількість кошиків не перевищувала наявних. Зауважимо, що в підході LALB кожне ядро працює з однаковою швидкістю, а це означає, що всі доступні контейнери (активні ядра) мають однакову ємність. У ТВ-підході, однак, різні ядра можуть мати різну швидкість, тому ми пакуємо завдання в ряд контейнерів з різною ємністю контейнерів.

Також ключовою особливістю масштабування напруги та частоти з міграцією є функція динамічного масштабування напруги та частоти (VFSM) (рисунок 2.6). Вона складається з системи безперервного моніторингу для обчислення швидкості та температури кожного ядра в мультипроцесорі. Після завершення розподілу завдань VFSM буде працювати в циклічному режимі з наперед визначеним значенням епохи, щоб зберегти температуру та продуктивність.

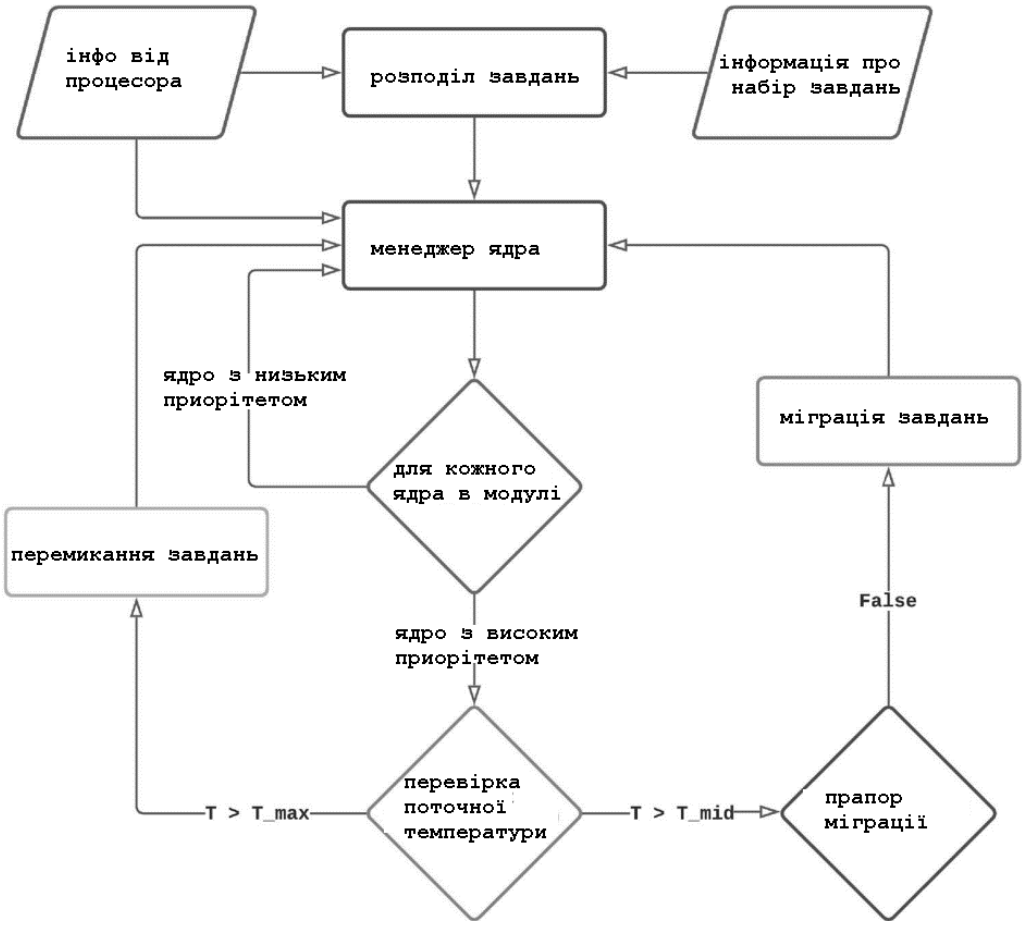


Рисунок 2.6 - Абстрактна блок-схема пропонованого методу

Висновки до розділу 2:

У розділі проведено дослідження фізично-інформованого теплового моделювання, описано режими яке може відображати дане моделювання враховуючи динамічну або статичну варіацію граничних умов та внутрішніх джерел енергії, розроблено і описано проекцію теплової задачі для багатоядерної системи в складі IoT модуля. Також в даному розділі розроблено алгоритм планування завдань які виконуються на процесорі на основі вищезгаданої моделі. Крім цього розроблено алгоритму планування з урахуванням теплового режиму в склад якого входять підалгоритми:

- управління станом ядра з урахуванням теплового режиму;
- вибору завдань;
- розподілу завдань між основними завданнями.

Оцінено складність виконання алгоритму планування з урахуванням теплового режиму. Описано принцип розподілу завдань за допомогою пакування контейнерів

3 РЕАЛІЗАЦІЯ МЕТОДУ ЗНИЖЕННЯ ПІКОВИХ ТЕМПЕРАТУР В IOT ПРИСТРОЯХ ЗА ДОПОМОГОЮ МІГРАЦІЇ ТА ЗАМІНИ ЗАВДАНЬ

3.1 Модель оцінки методів управління ресурсами

Для оцінки методів управління ресурсами IoT модулів на базі багатоядерних платформ, використовується загальна експериментальна установка, в той час, як використовуються різні гомогенні та гетерогенні багатоядерні архітектури.

В однорідних архітектурах використовуються позачергові (O3) альфа-ядра. В той час як в гетерогенних архітектурах використовуються чотири типи ядер: O3 Alpha ядра, прості Alpha ядра, впорядковані ядра ARM та O3 ARM-A15. На додаток до цих ядер, як прискорювачі використовуються щільно з'єднані процесорні масиви (Tightly-Coupled Processor Arrays, TCPA). Огляд експериментального середовища показано на рисунку 3.1.

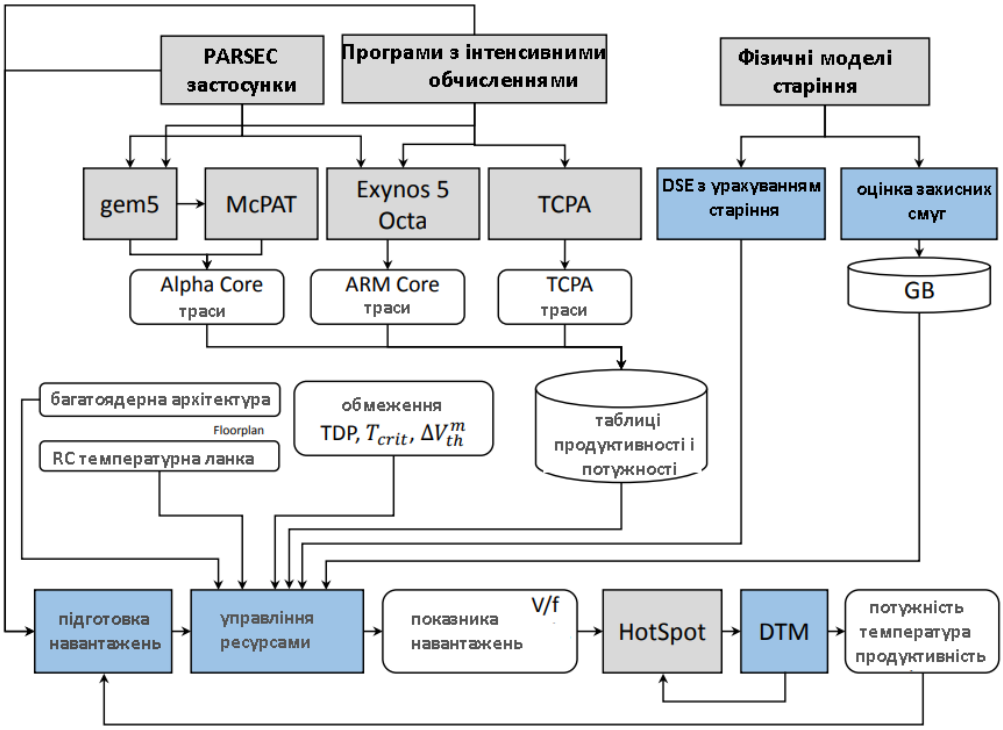


Рисунок 3.1- Огляд експериментальної бази, використаної для оцінки методів управління ресурсами

Ця експериментальна база інтегрує симулятори gem5 [42] та McPAT [42] для моделювання ядер Alpha з використанням повносистемного режиму.

Додатково (для гетерогенної архітектури) вона інтегрує реальні вимірювання для ядер ARM та результати постсинтезу для TCPA з RTL-компілятора Cadence Encounter.

Для бенчмарків використано реалістичні багатопотокові додатки з набору тестів PARSEC. Цей набір було розроблено як репрезентативний для додатків наступного покоління зі спільною пам'яттю для багатоядерних архітектур. Він містить різні програми з різних галузей, таких як комп'ютерний зір, кодування відео, фінансова аналітика тощо. Ці програми можуть виконуватися на ядрах Alpha та ARM.

Крім додатків PARSEC, використовуються додаткові обчислювально інтенсивні додатки "SAD", "Edge Detection", "FIR", "Optical Flow", "Matrix Multiplication" та "Harris Corner", які можуть виконуватися або на ядрах Alpha та ARM.

Для симуляцій з gem5 та McPAT кожен додаток спочатку виконується в gem5 на різних типах ядер Alpha (O3 та простих) та для всіх можливих рівнів V/f та різної кількості паралельних потоків.

З вихідної статистики gem5 будуються таблиці продуктивності додатків, як показано на рисунку 3.2.

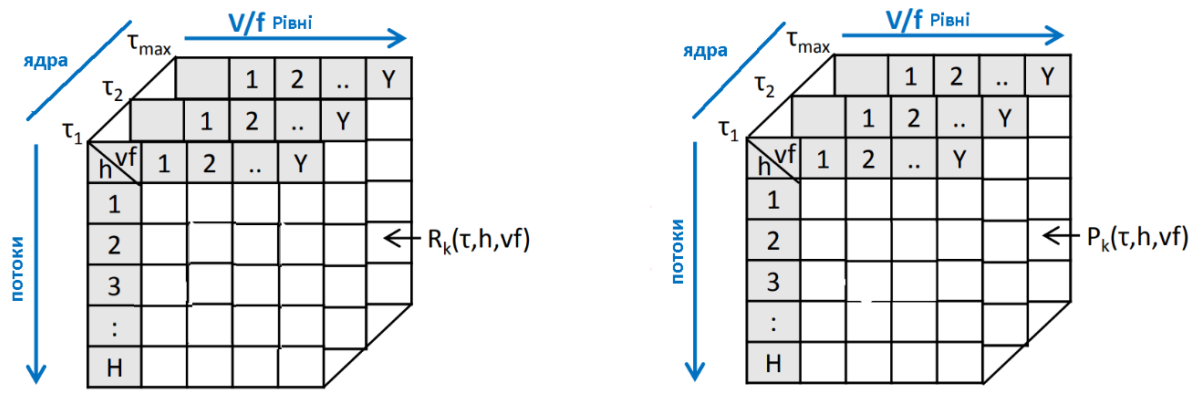


Рисунок 3.2 - Таблиці продуктивності та потужності для додатків.

Також витягується відповідна статистика, необхідна для McPAT. Потім виконується McPAT, щоб оцінити площі та енергоспоживання ядер та інших блоків архітектури, таких як кеші та контролери пам'яті.

Ці статистичні дані використовуються для побудови таблиць енергоспоживання додатків, тоді як оцінені площі різних блоків використовуються для побудови поверхових планів використаних архітектур.

Для вимірювання ядер ARM на платформі кожен додаток виконувався на різних типах ядер ARM (O3 та in-order) для всіх можливих рівнів V/f та різної кількості паралельних потоків. Під час виконання зчитувались показники датчиків живлення ядер, а в кінці вимірюється загальний час виконання.

Для запуску додатків на TCPA коди генеруються за допомогою PARO [44], де проблема планування та зв'язування в умовах обмежених ресурсів вирішується за допомогою змішаного цілочисельного лінійного програмування (MILP),

PARO містить інтерфейс, орієнтований на програмовані апаратні прискорювачі. Тут для відображення циклічних програм реалізовано методи символічного розпаралелювання як перетворення компілятора для розбиття та планування гнізд циклів з використанням параметризованих розмірів.

Траси потужності та продуктивності для TCPA отримано на основі результатів постсинтезу за допомогою RTL-компілятора Cadence Encounter.

Подібно до трас для ядер Alpha, вимірювання для ядер ARM та траси для TCPA також будуть використані для побудови відповідних таблиць потужності та продуктивності для додатків.

Запропоновані методи управління ресурсами потребують оцінки температур в процесі прийняття рішень. Тому використовується теплова модель на основі RC-термічної мережі.

Для отримання необхідних метрик прийнятої RC-термічної мережі, таких як B і G у рівнянні (12).

$$T_{steady} = BP + T_{amb}BG \tag{12}$$

Значення у векторі-стовпчику Tsteady = [Ti]Z×1 - це стаціонарні температури на теплових вузлах. Матриця B = [bi,j]Z×Z є оберненою до матриці B і містить величину теплового внеску теплових вузлів.

В побудові бази вхідних дани використовується симулятор HotSpot з конфігураціями за замовчуванням, а саме Товщина мікросхеми 0,15 мм, теплопровідність кремнію 100 Вт м-К, питома теплоємність кремнію 1,75 - 106 Дж м3-К, радіатор 6×6 см і товщиною 6,9 мм, конвекційна ємність радіатора 140,4 Дж К, конвекційний опір радіатора 0. 1 КВт, теплопровідність радіатора і теплорозподільника 400 Вт м-К, питома теплоємність радіатора і теплорозподільника 3,55-106 Дж м3-К, теплорозподільник розміром 3×3 см і товщиною 1 мм, товщина матеріалу інтерфейсу 20 мкм, теплопровідність матеріалу інтерфейсу 4 Вт м-К, питома теплоємність матеріалу інтерфейсу 4 - 106 Дж м3-К. Температура навколишнього середовища становить 45 °C.

Крім того, ця структура включає в себе моделювання старіння, описане на Рисунку 3.3.

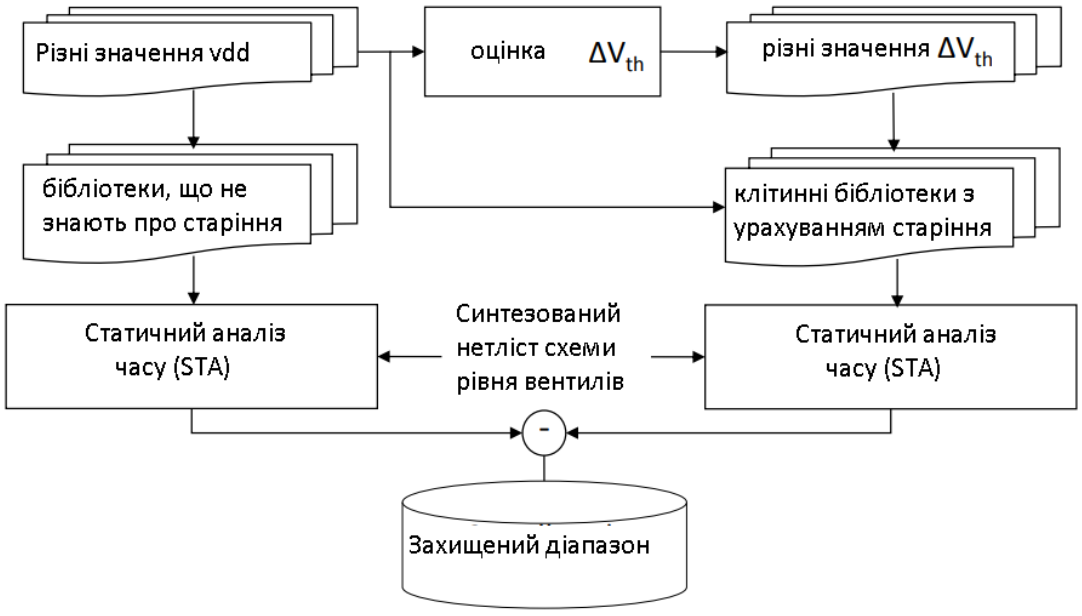


Рисунок 3.3 - Оцінка захисних смуг синхронізації з урахуванням різних значень Vdd

Для того, щоб уможливити дослідження простору проектування старіючого програмного.

Крім того, процес оцінки захисної смуги (Рисунок 3.3) включений, щоб дозволити методам управління ресурсами застосовувати необхідні захисні смуги для підтримки надійності.

Всі попередні кроки необхідно виконати один раз (під час проектування) перед запуском циклу управління. Зокрема, в цій системі використовується цикл управління, який запускається з інтервалом у 10 мс, подібно до епохи планування в Linux.

У цьому циклі управління запропоновані методи та кандидати для порівняння (обидва представлені у цій системі як "Управління ресурсами") спочатку приймають рішення про розподіл ресурсів між додатками поточного робочого навантаження та зіставлення цих додатків з ядрами. Результуюча перехідна температура буде розрахована за допомогою симулятора HotSpot. Ми припускаємо, що стандартна методика DTM реалізована на мікросхемі, подібно до комерційних процесорів [47]. Таким чином, якщо перехідна температура будь-якого ядра перевищує критичну температуру Tcrit, DTM спрацьовує для охолодження ядер. DTM має інтервал контролю 1 мс, оскільки передбачається, що він має бути реалізований на апаратному рівні для забезпечення швидкої реакції на теплові порушення. Таким чином, на кожному інтервалі контролю DTM досліджує перехідні температури і знижує рівні V/f до мінімального рівня, якщо температура перевищує Tcrit. В іншому випадку, якщо вона знову опускається нижче Tcrit, DTM знову підвищує рівні V/f до початкового рівня V/f, визначеного методом управління ресурсами.

На рисунку 3.4 показано деталі технології DTM, реалізованої в Cortex A53 відповідно до його специфікації. Важливо зазначити, що не всі з представлених методів управління ресурсами обов'язково будуть виконуватися на кожному часовому інтервалі. Деякі з них будуть виконуватися тільки при зміні робочого

навантаження, деякі з них приймають тільки статичне робоче навантаження і тому будуть виконуватися тільки один раз на початку циклу управління.

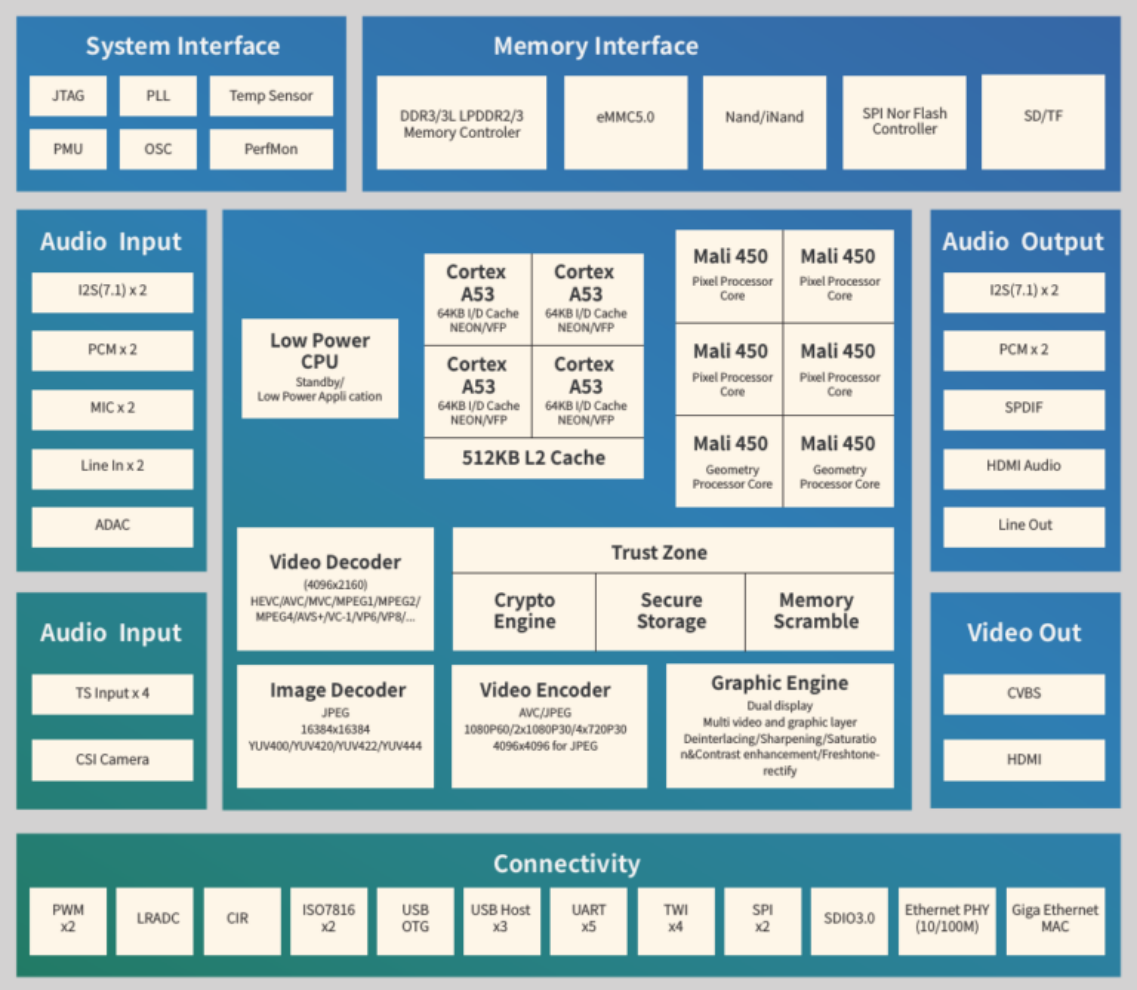


Рисунок 3.4 – Структура Cortex A53

3.2 Експериментальна оцінка та аналіз

Експериментальна оцінка алгоритму була виконана на симуляторі, названому багатоядерним симулятором планування в реальному часі, MCRTsim (Рисунок 3.5). В експериментах було використано реалістичне середовище на основі процесора PXA270 на базі технології XScale від Marvell та неідеальної платформи DVS.

Процесор Cortex A53 забезпечує шість рівнів частоти напруги, зазначених у таблиці 1. За допомогою симулятора MCRTsim було створено двоядерний процесор з налаштуванням його доступних тактових частот. Основним показником продуктивності, було енергоспоживання завдань, назване Energy_Consum.

Припускаючи, що s(t) - це швидкість процесора в момент часу t, енергоспоживання Energy_Consum можна визначити за формулою

$$\int_0^{simTime} PC(s(t))dt \quad (13)$$

де simTime - це час симуляції.

Таблиця 3.1 - Рівні частоти напруги Cortex A53

Параметр	Рівень 1	Рівень 2	Рівень 3	Рівень 4	Рівень 5	Рівень 6
Напруга	1.55	1.45	1.35	1.25	1.15	0.90
Частота	624	520	416	312	208	104
Активне енергоспоживання	925	747	570	390	279	116
Споживання енергії в режимі очікування	260	222	186	154	129	064

Після того, як завдання закріплені за процесорами, обирається схема розподілу швидкостей, щоб зменшити енергоспоживання, зберігаючи при цьому реалістичність. Алгоритм алгоритму планування з урахуванням теплового режиму, спочатку використовується для виконання завдань на низькій швидкості, а потім миттєво перемикається на високу швидкість, коли завдання блокуються. При використанні алгоритму для планування завдань, DS регулює рівень низької та високої швидкості на основі умови достатньої планувальності.

3.2.1 Налаштування моделювання

Було використано різні робочі навантаження, випадково згенеровані набори завдань. Значення періоду виконання завдань були згенеровані рівномірно, щоб отримати короткі завдання (10~50) мс, середні завдання (50~100) мс та довгі завдання (100~500) мс.

Найгірший час обчислення завдань у трьох групах становив (1~10), (1~20) та (1~100) мс відповідно. Період виконання завдання та найгірший обсяг обчислень були обрані випадковим чином з відповідних діапазонів для кожного робочого навантаження. Кожен набір завдань складався з 5-20 завдань.

У цьому дослідженні кількість спільних ресурсів була скоригована до 4 і 6, щоб забезпечити достатню конкуренцію між завданнями. Кількість ресурсів, до яких звертається задача, вибиралася випадковим чином від 1 до 4. Тривалість і розташування критичних ділянок у кожній задачі вибиралися випадковим чином. Межа використання для EDF становить n $\sum$ i=1 Ci Pi ≤ 1, де n - кількість задач. У таблиці 2 наведено параметри для набору з 10 завдань з коротким періодом.

Таблиця 3.2 Приклад набору параметрів для десяти коротких завдань.

i	1	2	3	4	5	6	7	8	9	10
A_i	0	0	1	3	0	1	0	2	2	0
P_i	30	27	43	45	49	40	48	50	47	39
C_i	2	1	3	6	4	3	4	7	1	1
U_i	0.067	0.037	0.069	0.133	0.082	0.075	0.083	0.14	0.021	0.026
$Z_{i,q}$	Z1,3 Z1,4	Z2,5 Z2,4	$Z_{3,2}$ $Z_{3,1}$	$Z_{4,2}$	$Z_{5,1}$ $Z_{5,2}$ $Z_{5,3}$	$Z_{6,4}$ $Z_{6,2}$	-	-	$Z_{9,5}$ $Z_{9,1}$ $Z_{9,3}$	

3.3 Результати оцінювання

Оцінка виконувалась за методологією, описаною вище для алгоритму планування з урахуванням теплового режиму. Щоб виконати оцінку за допомогою нашої POD-моделі, спочатку потрібно визначити відповідну кількість режимів POD, які будуть використовуватися для прогнозування під час планування. З цією метою модель POD навчається для процесора і оцінюється на розкладі завдань, щоб змоделювати, як вона буде працювати при прогнозуванні температури процесора під час виконання розкладу.

Щоб отримати істинні дані для порівняння прогнозів POD, використовується FEniCS (FEM DNS) для розрахунку температурного профілю того ж розкладу.

Порівняння проводиться з використанням найменшої квадратичної похибки (LSE) в якості метрики. LSE для кожного кроку розраховується за допомогою рівняння (14), де Nx, Ny та Nz - кількість кроків сітки теплового моделювання в напрямках x, y та z відповідно; Ti - температура в i-й точці простору, розрахована за допомогою FEniCS; Pi - температура в i-й точці простору, прогнозована за допомогою POD.

$$LSE = 100 \cdot \sqrt{\frac{\sum_{i=0}^{N_x \cdot N_y \cdot N_z} (T_i - P_i)^2}{\sum_{i=0}^{N_x \cdot N_y \cdot N_z} T_i^2}} \tag{14}$$

Це дає результати прогнозування за допомогою LSE у часі, як показано на рисунку 9.

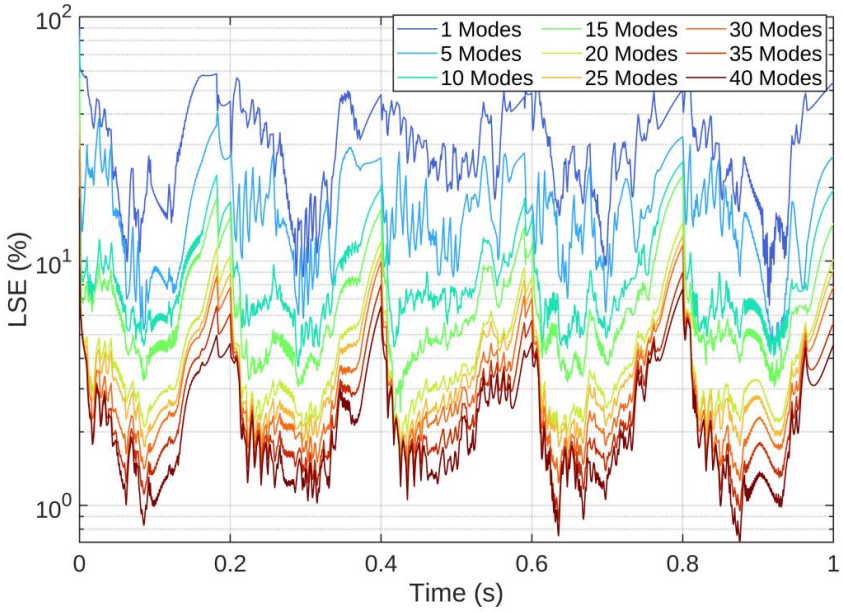


Рисунок 3.5 – Найменша квадратична похибка всього чіпа з часом на кількість режимів

Ключовою особливістю цих вимірювань похибок є те, що, хоча вони змінюються з часом, вони не мають тенденції до зростання. Це свідчить про те, що прогнози моделі POD залишаються точними протягом дуже тривалого періоду часу.

Середні значення найменшої квадратичної похибки за певну кількість режимів показані на рисунку 3.6

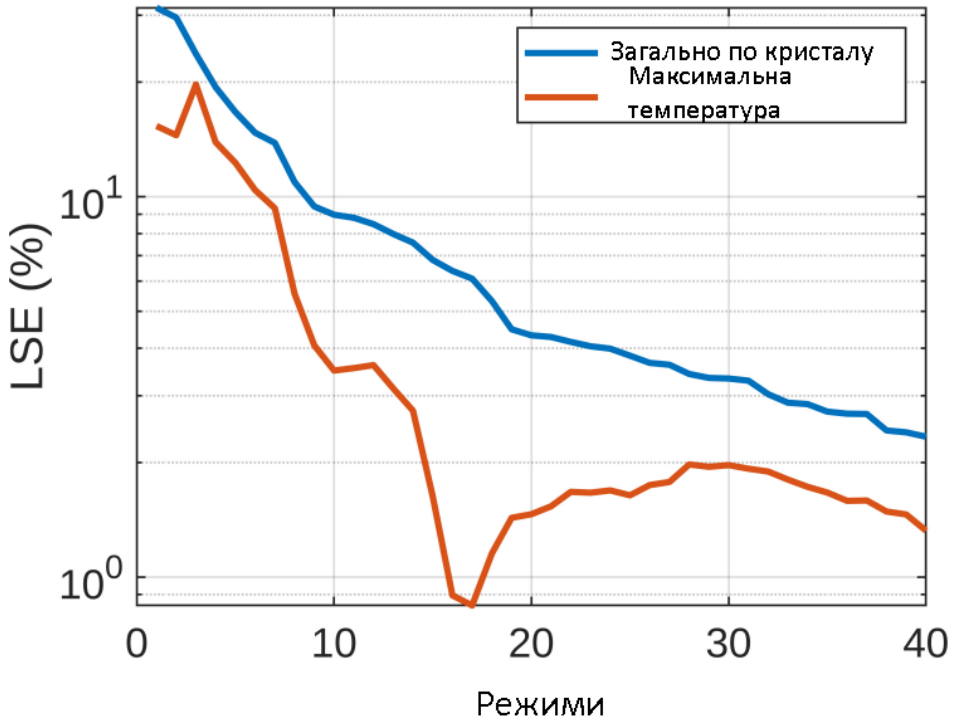


Рисунок 3.6 – Середнє значення найменшої квадратичної похибки на кількість POD режимів за 1 секунду редукції

На рисунку 3.6 показано LSE максимальної прогнозованої температури в активному шарі мікросхеми. Це ключове значення, оскільки найбільше цікава є пікова температура мікросхеми під час планування.

Виходячи з графіків 3.5 і 3.6, прогнозування для побудови 30 режимів, щоб збалансувати точність моделі та обчислювальні накладні витрати.

На рисунку 3.7 показано деякі характеристики графіків для різних комбінацій ТС і ТН. Рисунок 3.7а та рисунок 3.7б показують частоту призначення завдань для набору оцінювання з 4-х завдань та набору оцінювання з 8-ми завдань відповідно.

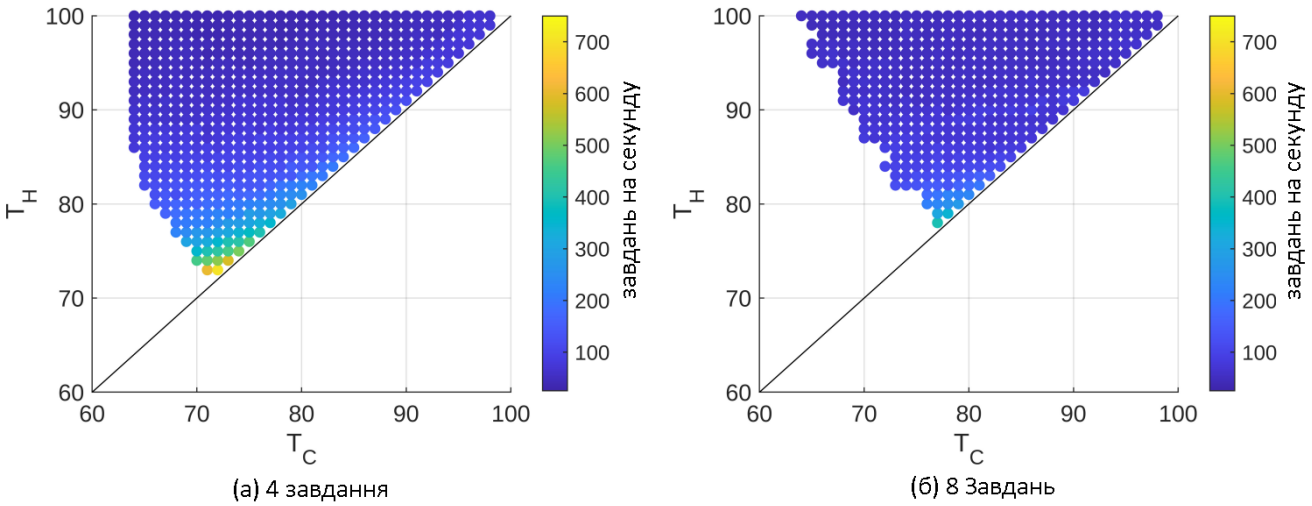


Рисунок 3.7 – Частота призначення завдань

На рисунку 3.8 показано приклад температури процесора, розрахованої за допомогою FEniCS під час виконання розкладу, порівняно з температурою процесора, яку прогнозував POD під час складання розкладу. Для цього випадку TH = 75◦C і TC = 70◦C. Ліва частина рисунку 3.8 показує перші 200 мс побудови графіка, а права частина показує прогнози між 1800 мс і 2 с.

Прогнози POD точно відповідають максимальній температурі з FEniCS. Крім того, видно, що кожне ядро поводить себе відповідно до станів процесора. Ядра, які перевищили TH, простоюють, щоб охолонути, поки їх температура не стане нижчою за TC .

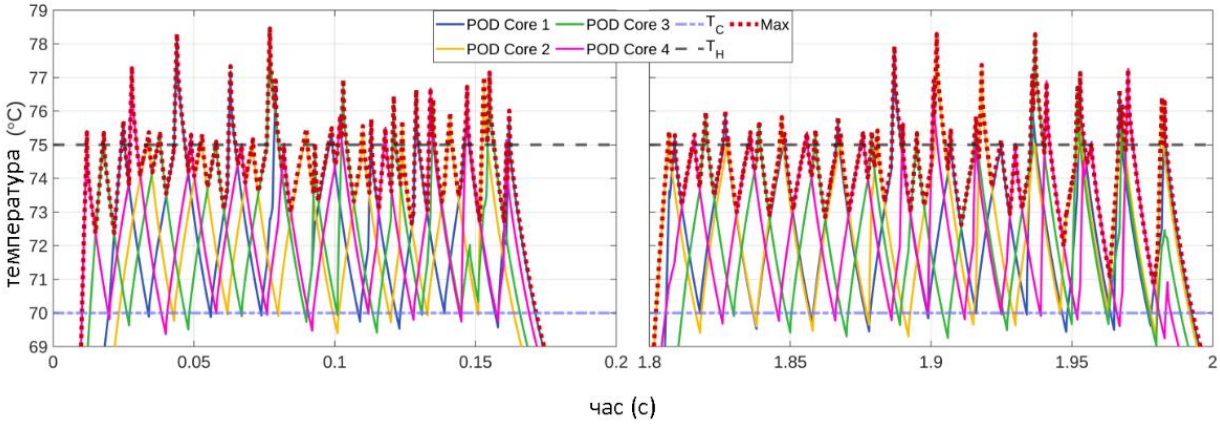


Рисунок 3.8 – Прогнози під час побудови розкладу

3.3.1 Налаштування та результати моделювання

Для імітаційного підходу архітектура чотирьохядерного контролера Cortex A53 була спроектована в gem5, симуляторі, розробленому для дослідження архітектури комп'ютерних систем.

У таблиці 3 наведено параметри системи, що використовуються в gem5 для проведення початкового моделювання.

Таблиця 3.3 - Бенчмарки PARSEC

Набір A	
Компонент	Параметр
Тип MCU	Cortex A53
Число ядер	4
Кеш L1	32
Кеш L2	1
Тип пам’яті	2048

На рисунку 3.9 зображено теплову карту гарячих точок, згенеровану під час застосування запропонованих методів. Рисунок 3.9(а) демонструє температуру ядер до виконання VFSM, де ядра 1 і 4 отримали завдання з високим пріоритетом, а ядра 2 і 3 отримали завдання з низьким пріоритетом. На рисунку 3.9(б) зображено теплову карту кластера після застосування VFSM зі зменшеною кількістю гарячих точок у кластері ядер.

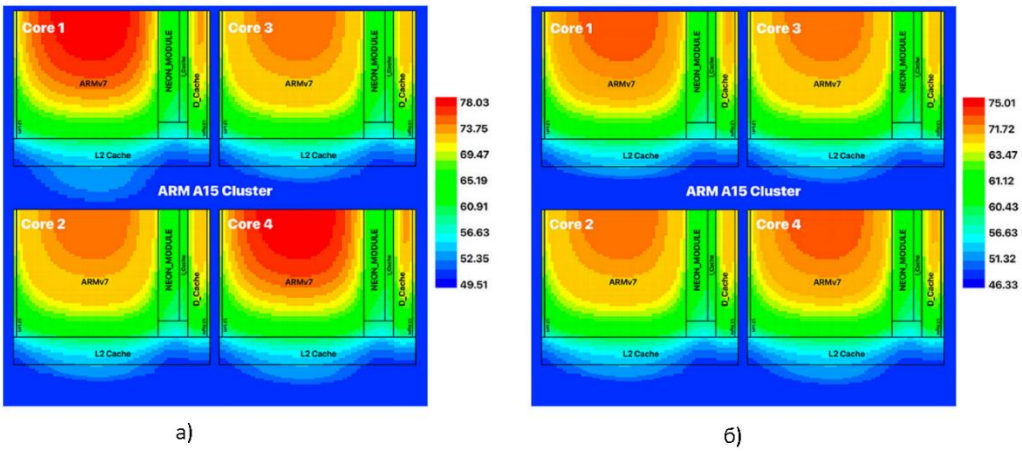


Рисунок 3.9 – Результати симуляції гарячих точок кластера ARM

Температурні показники першої пари ядер під час виконання запропонованих методів у симуляції зображено на рисунку 3.10. Як видно з рисунку 3.10, на 800-му інтервалі часу ядру 1 присвоєно високий пріоритет, а ядру 2 - низький, відповідно до пріоритету завдань, покладених на кожне з ядер.

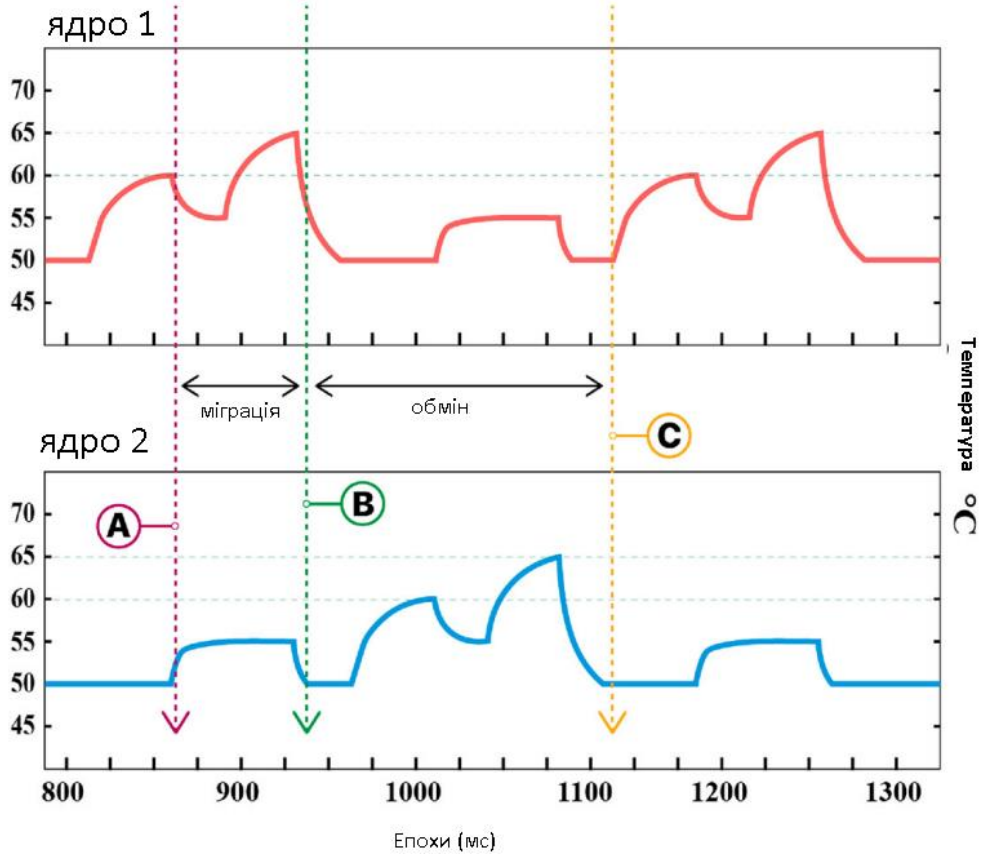


Рисунок 3.10 – Результати моделювання вихідної пари ядер за допомогою запропонованого методу для екстремальної ситуації

Ядро 1 в точці (A) досягає допустимого порогу середньої температури, після чого диспетчер ядер перевіряє, чи піднято прапорець міграції завдань для ядра 1, і в цьому прикладі, оскільки прапорець не було піднято, він продовжує виконувати міграцію завдань.

Запропонований метод міграції завдань призводить до падіння швидкості ядра 1, водночас збільшуючи швидкість ядра 2 настільки, щоб витримати терміни виконання перенесеної частини завдання. Між точками (A) і (B) можна спостерігати коливання температури двох ядер.

Температура високопріоритетного ядра постійно контролюється менеджером ядер, і коли вона досягає максимально допустимої температури, менеджер ядер виконує перестановку завдань, оскільки міграція завдань між парою ядер вже відбулася.

Це явище можна спостерігати в точці (B), де міграція завдань повертається назад, і відбувається обмін завданнями. Високопріоритетне робоче навантаження переміщується на ядро 2, а низькопріоритетне - на ядро 1. Пріоритети цих ядер відповідно міняються місцями, де ядро 1 тепер є низькопріоритетним ядром, а ядро 2 - високопріоритетним ядром. Зміни температури цих двох ядер можна побачити між точками (B) і (C).

Висновки до розділу 3:

У цьому розділі представлено модель оцінки методів управління ресурсами, проведено експерементальну оцінку та аналіз методів, проведено налаштування моделювання, отримано результати експериментів та проведено їх оцінювання. Також представлено новий підхід до зменшення гарячих точок, градієнтів та надмірного розсіювання потужності ядер в контролері модуля IoT, зберігаючи при цьому достатню пропускну здатність.

Запропонований метод, є двокомпонентним підходом до вирішення описаних проблем за допомогою міграції та заміни завдань в режимі онлайн.

ВИСНОВКИ

При виконанні роботи був проведеней аналіз базових принципів розсіювання потужності мікропроцесорних та мікроконтролерних систем та проблеми з якими стикаються розробники.

Під час цієї роботи проведено аналіз основних принципів розподілу потужності в мікропроцесорних та мікроконтролерних системах, а також виявлено проблеми, з якими стикаються розробники. Було досліджено принципи управління енергоспоживанням під час виконання завдань, динамічного регулювання напруги/частоти та контролю живлення, а також вивчено поведінку процесів при динамічній зміні живлення пристрою.

Також було розглянуто методи планування завдань, враховуючи температурні умови, та виявлено проблеми, з якими стикаються розробники та виробники мікроконтролерів та процесорів для модулів Інтернету речей. Було здійснено аналіз методів попереднього моделювання завдань для досягнення енергоефективності та використання методів моделювання теплових процесів у багатоядерних системах, що входять до складу модулів Інтернету речей.

Також проведено дослідження фізично-заснованого моделювання тепла, в якому описано режими, що можуть відтворюватися цією моделлю, враховуючи зміни у граничних умовах та внутрішніх джерелах енергії, будь то динамічні або статичні. Також була розроблена та описана проекція теплової задачі для системи з багатьма ядрами, що входять в склад модулів Інтернету речей.

Крім цього було розроблено алгоритм планування завдань для виконання на процесорі, що базується на зазначеній моделі. Крім того, був розроблений алгоритм планування, який враховує тепловий режим і включає в себе підалгоритми:

- управління станом ядра з огляду на тепловий режим;
- вибір завдань;
- розподіл завдань між основними завданнями.

Була проведена оцінка складності виконання алгоритму планування з урахуванням теплового режиму. Також було описано принцип розподілу завдань за допомогою пакування контейнерів.

Крім цього розглянуто модель для оцінки стратегій управління ресурсами, здійснено експериментальну перевірку та аналіз методів, налаштовано моделювання, проаналізовано результати експериментів та їхню оцінку. Також представлено новий підхід до зниження гарячих зон, градієнтів та надмірної розсіювання потужності ядер у контролері модуля Інтернету речей, при цьому забезпечуючи достатню пропускну здатність.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Józefowska, Joanna, et al. Survey on Optimization Models for Energy-Efficient Computing Systems. Energies, 2022, 15.22: 8710.
2. Sachdeva, Ashish; TOMAR, V. K. Design of a stable low power 11-T static random access memory cell. Journal of circuits, Systems and Computers, 2020, 29.13: 2050206.
3. Muralidhar, Rajeev; BOROVICA-GAJIC, Renata; BUYYA, Rajkumar. Energy efficient computing systems: Architectures, abstractions and modeling to techniques and standards. ACM Computing Surveys (CSUR), 2022, 54.11s: 1-37.
4. Cao, Wei, et al. The future transistors. Nature, 2023, 620.7974: 501-515.
5. Hossain, Md Shazzad. Power Management Techniques for Ultra-Low Voltage Integrated Circuits. Drexel University, 2021.
6. Puche LARA, José. Novel cache hierarchies with photonic interconnects for chip multiprocessors. 2021. PhD Thesis. Universitat Politècnica de València.
7. Schall, David, et al. "Lukewarm serverless functions: characterization and optimization." Proceedings of the 49th Annual International Symposium on Computer Architecture. 2022.
8. Gul, Waqas; SHAMS, Maitham; AL-KHALILI, Dhamin. SRAM Cell Design Challenges in Modern Deep Sub-Micron Technologies: An Overview. Micromachines, 2022, 13.8: 1332.
9. Acharya, Ishaan. Energy consumption and architecture and computer technology subjects. International Journal of Innovative Research in Engineering, 2022, 3.4: 106-112.
10. Quan, Zhe, et al. Task scheduling for energy consumption constrained parallel applications on heterogeneous computing systems. IEEE Transactions on Parallel and Distributed Systems, 2019, 31.5: 1165-1182.
11. Bamas, Étienne, et al. Learning augmented energy minimization via speed scaling. Advances in Neural Information Processing Systems, 2020, 33: 15350-15359.

12. Marwedel, Peter. Embedded system design: embedded systems foundations of cyber-physical systems, and the internet of things. Springer Nature, 2021.
13. Kumar, Neetesh; Vidyarthi, Deo Prakash. A novel energy-efficient scheduling model for multi-core systems. Cluster Computing, 2021, 24.2: 643-666.
14. Ali, Haider, et al. A survey on system level energy optimisation for MPSoCs in IoT and consumer electronics. Computer Science Review, 2021, 41: 100416.
15. Haj-Yahya, Jawad, et al. SysScale: Exploiting multi-domain dynamic voltage and frequency scaling for energy efficient mobile processors. In: 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2020. p. 227-240.
16. Yu, Zheqi, et al. Energy and performance trade-off optimization in heterogeneous computing via reinforcement learning. Electronics, 2020, 9.11: 1812.
17. Shahid, Arsalan, et al. Energy predictive models of computing: theory, practical implications and experimental analysis on multicore processors. IEEE Access, 2021, 9: 63149-63172.
18. Saez, Juan Carlos; PRIETO-MATIAS, Manuel. Evaluation of the Intel Thread Director technology on an Alder Lake processor. In: Proceedings of the 13th ACM SIGOPS Asia-Pacific Workshop on Systems. 2022. p. 61-67.
19. Ramesh, Srinivasan, et al. Understanding the impact of dynamic power capping on application progress. In: 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS). IEEE, 2019. p. 793-804.
20. Mandal, Sumit K., et al. An energy-aware online learning framework for resource management in heterogeneous platforms. ACM Transactions on Design Automation of Electronic Systems (TODAES), 2020, 25.3: 1-26.
21. Safari, Sepideh; Hessabi, Shaahin; Ershadi, Ghazal. LESS-MICS: A low energy standby-sparing scheme for mixed-criticality systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39.12: 4601-4610.

22. Wright, John Charles, et al. A dual-core risc-v vector processor with on-chip fine-grain power management in 28-nm fd-soi. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2020, 28.12: 2721-2725.
23. Iskandar, Veronia; Salama, Cherif; Taher, Mohamed. Dynamic thread mapping for power-efficient many-core systems under performance constraints. Microprocessors and Microsystems, 2022, 93: 104614.
24. Da Silva, Alzemiro Lucas; Del Mestre Martins, André Luís; Moraes, Fernando G. Mapping and Migration Strategies for Thermal Management in Many-Core Systems. In: 2020 33rd Symposium on Integrated Circuits and Systems Design (SBCCI). IEEE, 2020. p. 1-6.
25. Pérez Rodríguez, Javier, et al. B-TSP: An Advanced Power Safe Management Strategy for modern Multi-core Platforms under Thermal-Aware Design. In: Proceedings of the 31st International Conference on Real-Time Networks and Systems. 2023. p. 34-44.
26. Yao, Yuan. Game-of-Life Temperature-Aware DVFS Strategy for Tile-Based Chip Many-Core Processors. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2023, 13.1: 58-72.
27. Wolff, Willy. It is too hot in here! A performance, energy and heat aware scheduler for Asymmetric multiprocessing processors in embedded systems. 2023. PhD Thesis. Lancaster University.
28. Zhou, Junlong, et al. Security-critical energy-aware task scheduling for heterogeneous real-time MPSoCs in IoT. IEEE Transactions on Services Computing, 2019, 13.4: 745-758.
29. Janna, William S. Engineering heat transfer. CRC press, 2018.
30. Hameed, Ahmad Raza, et al. Energy-and performance-aware load-balancing in vehicular fog computing. Sustainable Computing: Informatics and Systems, 2021, 30: 100454.

31. Chen, Yu, et al. Atmem: Adaptive data placement in graph applications on heterogeneous memories. In: Proceedings of the 18th ACM/IEEE International Symposium on Code Generation and Optimization. 2020. p. 293-304.
32. Kaul, Yaniv. Disabling in-memory caching of a virtual machine during migration. U.S. Patent No 10,635,477, 2020.
33. LI, Xin, et al. Accurate on-chip temperature sensing for multicore processors using embedded thermal sensors. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2020, 28.11: 2328-2341.
34. Kong, Joonho; CHUNG, Sung Woo; SKADRON, Kevin. Recent thermal management techniques for microprocessors. ACM Computing Surveys (CSUR), 2012, 44.3: 1-42.
35. LI, Tiantian, et al. Minimizing temperature and energy of real-time applications with precedence constraints on heterogeneous mpsoc systems. Journal of Systems Architecture, 2019, 98: 79-91.
36. Da Silva, Alzemiro Lucas; Martins, André Luís del Mestre; Moraes, Fernando Gehm. Fine-grain temperature monitoring for many-core systems. In: Proceedings of the 32nd Symposium on Integrated Circuits and Systems Design. 2019. p. 1-6.
37. Mohammed, Mohammed Sultan, et al. Temperature-aware task scheduling for dark silicon many-core system-on-chip. In: 2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO). IEEE, 2019. p. 1-5.
38. Jiang, Lin; LIU, Yu; Cheng, Ming-C. Fast accurate full-chip dynamic thermal simulation with fine resolution enabled by a learning method. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022.
39. LI, Hui, et al. A Thermal Twin Modeling Method of Press Pack IGBT Based on Power Loss. IEEE Transactions on Electron Devices, 2022, 69.12: 6922-6928.
40. Vinuesa, Ricardo; Brunton, Steven L. Enhancing computational fluid dynamics with machine learning. Nature Computational Science, 2022, 2.6: 358-366.
41. Li, Tiantian, et al. TA-MCF: Thermal-aware fluid scheduling for mixed-criticality system. Journal of Circuits, Systems and Computers, 2019, 28.02: 1950029.

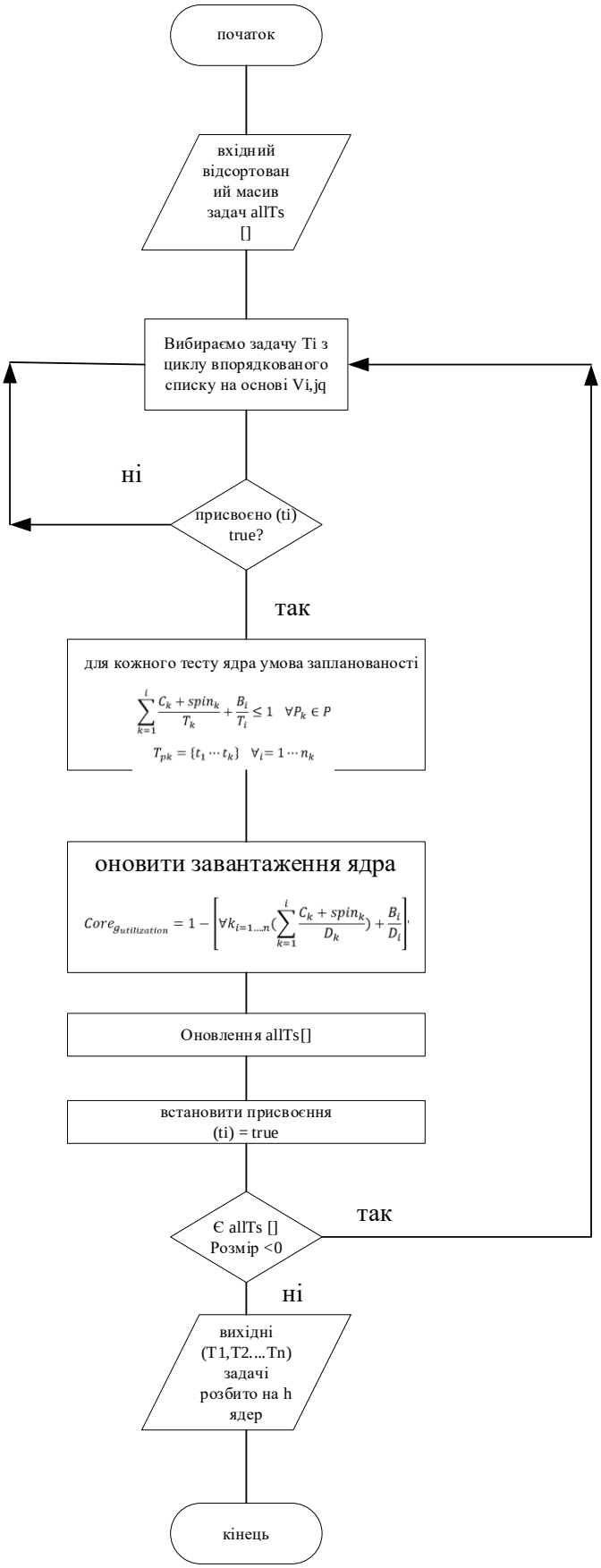
42. Rupanetti, Dulana; Salamy, Hassan. Thermal and energy-aware utilisation management on MPSoC architectures. International Journal of Parallel, Emergent and Distributed Systems, 2021, 36.5: 449-469.
43. LAI, Yi-Hsiang, et al. Programming and synthesis for software-defined FPGA acceleration: status and future prospects. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 2021, 14.4: 1-39.
44. Fickenscher, Jörg; Hannig, Frank; Teich, Jürgen. DSL-based acceleration of automotive environment perception and mapping algorithms for embedded CPUs, GPUs, and FPGAs. In: Architecture of Computing Systems–ARCS 2019: 32nd International Conference, Copenhagen, Denmark, May 20–23, 2019, Proceedings 32. Springer International Publishing, 2019. p. 71-86.
45. Doweck, Jack, et al. Inside 6th-generation intel core: New microarchitecture code-named skylake. IEEE Micro, 2017, 37.2: 52-62.
46. Cacchiani, Valentina, et al. Knapsack problems—An overview of recent advances. Part II: Multiple, multidimensional, and quadratic knapsack problems. Computers & Operations Research, 2022, 143: 105693.
47. Intel Corporation, “Intel xeon phi processor datasheet,” https://ark.intel.com/products/94709/Intel-Xeon-Phi-Processor-7210F-16GB-1.30-GHz-64-core,December 2017.
48. Загальні рекомендації з підготовки, оформлення, захисту та оцінювання випускних кваліфікаційних робіт здобувачів вищої освіти першого «бакалаврського» і другого «магістерського» рівнів / За ред. доц. М.І. Шинкарика. Тернопіль: ТНЕУ, 2018. 67 с.
49. Комар М.П., Саченко А.О., Васильків Н.М. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за другим (магістерським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2021. 32 с.
50. Вишневський Д.С., Осолінський О.Р. « Планування завдань з урахуванням енергоспоживання для пристроїв Інтернету речей», Розвиток сучасної науки:

актуальні питання теорії та практики:матеріали IV Всеукраїнської студентської наукової конференції, м. Львів, 17 листопада, 2023 рік / ГО «Молодіжна наукова ліга». — Вінниця: ТОВ «УКРЛОГОС Груп», 2023. -289-292сс.

51. Вишневський Д.С., Осолінський О.Р. « Енергоефективне планування для пристроїв IoT в умовах теплових обмежень», Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення (випуск 82): матеріали Міжнародної наукової інтернет конференції, (м. Тернопіль, Україна, м. Ополе, Польща, 9-10 листопада 2023 р.) / редкол. : О. Патряк та ін. ГО “Наукова спільнота”, WSZIA w Opolu. Тернопіль : ФО-П Шпак В.Б. 2023 – 14-18сс/

ДОДАТОК А

Блок-схема алгоритму розподілу завдань



ДОДАТОК Б
АПРОБАЦІЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

УДК 001 (063)

Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення (випуск 82)*:* матеріали Міжнародної наукової інтернет-конференції, (м. Тернопіль, Україна, м. Ополе, Польща, 9-10 листопада 2023 р.) / редкол. : О. Патряк та ін. ГО "Наукова спільнота", WSZIA w Opolu. Тернопіль : ФО-П Шпак В.Б. 2023. 206 с. – ISSN 2522-932X

Збірник тез доповідей підготовлено за матеріалами Міжнародної наукової інтернет-конференції (випуск 82) 9-10 листопада 2023 р. на сайті www.konferenciaonline.org.ua

Наша адреса: Оргкомітет МНІК "Конференція онлайн"
а/с 797, м. Тернопіль 46005
тел. моб. 068 366 0 525
e-mail: inetkonf@ukr.net

URL Інтернет-конференції: http://www.konferenciaonline.org.ua/
ISSN 2522-932X

Література:

1. "Охорона природних ресурсів", В. А. Костіков, видавництво "Видавництво Академії наук України", 2013 рік.
2. "WordPress: від простого до складного", Андрій Ігнатенко, видавництво "ВІПОЛ", 2022 рік.
3. "Методи математичного моделювання", В. М. Глушков, О. М. Петренко, В. А. Субботін, видавництво "Київський університет", 2006 рік.

***Вишневський Дмитро Сергійович**, магістр,*
Західноукраїнський національний університет, м. Тернопіль

***Науковий керівник: Осолінський Олександр Романович**,*
доцент кафедри інформаційно-обчислювальних систем і управління,
Західноукраїнський національний університет, м. Тернопіль

ЕНЕРГОЕФЕКТИВНЕ ПЛАНУВАННЯ ДЛЯ ПРИСТРОЇВ IOT В УМОВАХ ТЕПЛОВИХ ОБМЕЖЕНЬ

Інтернет-адреса публікації на сайті:
http://www.konferenciaonline.org.ua/ua/article/id-1447/

Оскільки щільність потужності мікросхем інтегральних схем продовжує зростати, мінімізація енергії та зниження температури є двома найбільш важливими питаннями при проектуванні модулів IoT. Хоча мінімізація енергії тісно пов'язана зі зниженням температури, найбільш енергоефективний метод може бути не найефективнішим для виконання температурних обмежень і навпаки. У даній роботі представлено вирішення проблеми розподілу періодичних важких задач реального часу для потужних IoT пристроїв, щоб максимізувати загальну енергоефективність в умовах пікових температурних обмежень. На відміну від традиційного підходу балансування навантаження, тобто рівномірного розподілу робочого навантаження на мікросхему, пропонується стратегія термобалансування, тобто мінімізації теплового градієнту між активними ядрами, щоб підвищити загальну енергоефективність системи, особливо при жорстких температурних обмеженнях.

Для вирішення даного завдання спочатку потрібно визначити нижню межу енергоспоживання за, а потім перетворити задачу розбиття на задачу пакування контейнерів змінного розміру. Пропонована модель або підхід теплового балансування може значно покращити енергоефективність для IoT модулів які працюють на максимальній межі продуктивності.

ВСТУП

Стрімке зростання обчислювальних потреб та експоненціальне зростання енергоспоживання мікросхем ставить перед розробниками обчислювальних систем завдання мінімізації енергоспоживання та тепловідведення в екстремальних умовах роботи. Оптимізація енергоспоживання є однією з

першочергових цілей через економічний і екологічний тиски. З іншого боку, задля збільшення циклів місій з обмеженим терміном служби/розміром батареї, мінімізація енергії відіграє важливу роль у розробці мобільних систем, таких як мобільні телефони або пристроїв, IoT.

На температуру матриці сучасного багатоядерного процесора який входить в склад, наприклад, мікрокомп'ютерів для кінцевих модулів IoT сильно впливають технологічні вузли, які швидко масштабуються, що призводить до потенційної високої щільності живлення та виникнення гарячих точок. Високі температури негативно впливають на надійність системи, збільшуючи споживання енергії витоку, і навіть можуть безповоротно вивести з ладу апаратне забезпечення.

Сучасні процесори які випускаються для різних потреб, оснащені датчиками температури під час роботи, щоб запобігти перегріванню обладнання. Якщо температура чіпа перевищує попередньо визначені температурні пороги, то спрацьовує схема автоматичного вимкнення, що негативно впливає на продуктивність системи та спричиняє порушення роботи. Тому планування з урахуванням теплових обмежень стає критично важливою технікою для згладжування теплового градієнта і контролю пікової температури процесора при проектуванні.

У цій статті пропонується підвищення енергоефективності при плануванні набору важких періодичних задач реального часу на багатоядерній платформі при заданих обмеженнях на пікову температуру. Мінімізація енергоспоживання вже давно є проблемою яка інтенсивно вивчається та потребує різноманітних та новаторських рішень,. Через квадратичну залежність між динамічною потужністю та швидкістю процесора добре відомий принцип використання постійної швидкості процесора та максимального зниження швидкості процесора для зменшення енергоспоживання [1, 2]. Це робить балансування робочого навантаження хорошим евристичним підходом для мінімізації енергоспоживання

Хоча споживання енергії та температура тісно пов'язані між собою, як показано в [3], мінімізація споживання енергії та зниження температури не обов'язково завжди синхронізовані між собою. Існуючі методи енергозбереження, наприклад, [4, 5, 6], незважаючи на їх широке застосування, можуть бути не найефективнішими для досягнення найвищої енергоефективності в умовах теплових обмежень. Тому є проблема вибору найбільш енергоефективного режиму при заданих температурних обмеженнях для IoT модулів.

Таким чином, для максимізації енергоефективності при температурних обмеженнях потрібно вирішити наступні завдання:

1. Потрібно визначити та встановити теоретичну верхню межу енергоефективності,
2. Перетворити задачу розбиття багатоядерної задачі на задачу пакування контейнерів.

ОГЛЯД ВІДОМИХ РІШЕНЬ

Попередні проаналізовані роботи були зосереджені на динамічному енергозбереженні, яке можна мінімізувати, використовуючи найнижчу постійну швидкість виконання як на одноядерних [7], так і на багатоядерних архітектурах [8].

Тому була розроблена ідея «критичної швидкості» [9, 10], щоб збалансувати динамічне зменшення потужності та приріст енергії витоку для мінімізації загального споживання енергії.

Більшість розробників і дослідників не дотримуються однакових принципів оптимізації енергоспоживання. Наприклад, такі моменти спостерігаються в роботі [11] запропоновано стратегію відображення завдань для періодичних наборів і доведено, що в найгіршому випадку за допомогою пошуку розподілу завдань кожній обчислювальній одиниці можна доручити більше завдань з меншими витратами енергії. Пагані та ін. [12] запропонували мінімізацію енергії для набору періодичних завдань, використовуючи найнижчу напругу/частоту.

Автори в роботі [13] оптимізували пропускну здатність системи та енергоефективність, враховуючи варіації технологічного процесу та характеристики потужності/продуктивності додатків при максимально допустимих обмеженнях на потужність. Однак мало з підходів враховують температурні обмеження.

Існує декілька підходів до енергетичної оптимізації з температурними обмеженнями. Наприклад, беручи до уваги обмеження на пікову температуру, У роботі [14] запропоновано підхід, який по суті є метаевристичним алгоритмом пошуку, для мінімізації споживання енергії.

Авторами в роботі [15] запропонували розбиття задач як задачу про ранець для мінімізації енергії в умовах теплових обмежень. Їхній алгоритм складався з двох етапів: перший етап – мінімізація динамічного енергоспоживання між ядрами шляхом розподілу завдань між ядрами таким чином, щоб загальне динамічне енергоспоживання було мінімальним, а другий етап розподіляв можливе просідання між завданнями на кожному ядрі таким чином, щоб мінімізувати пікову температуру.

Автори роботи [16] оптимізували енергоефективність з урахуванням пам'яті в умовах енергетичних і теплових обмежень, що дозволяє краще вирішувати «пам'яттєвомісткі» завдання

МОДЕЛЬ СИСТЕМИ/ЗАДАЧІ

Якщо розглядати багатоядерну платформу Npl з N ядрами, то

$$Npl = \{\text{core}_i : i = 1, \text{---}, N\}, \qquad (1)$$

Кожне ядро може незалежно встановлювати різні швидкості або режими роботи, які характеризуються відповідною напругою живлення v та робочою частотою f. Оскільки $v \propto f$. Для зручності викладу використовується нормалізоване v для позначення швидкості обробки $v_{min} \le v \le v_{max}$.

Застосовуючи методи динамічного керування живленням (DPM), ядра, що простоюють (позначені через N_{dark}) і не мають завдань, можуть бути вимкнені, щоб уникнути витоків енергії [2, 13]. Інші активні ядра з завданнями належать до N_{active},

$$N_{active} = N \setminus N_{dark}, \tag{2}$$

Тоді можна вважати, що періодичне завдання складається з M завдань,

$$Tsk = \{\tau k : k = 1, \text{---}, M\}, \tag{3}$$

Кожна задача є однопоточною і визначається параметром інтервалу між надходженнями (періоду) та найгіршим часом виконання (WCET) при максимальній швидкості, тобто

$$\tau_k = \{Період_k, WCETk\}, \tag{4}$$

Можна припустити, що всі завдання надходять на початку періоду, а дедлайн кожного завдання дорівнює його періоду, тобто можна припустити, що обчислювальні ядра здатні забезпечити достатню кількість ресурсів, включаючи необхідну швидкість, пам'ять, кеш і пропускну здатність для виконання завдань. Для декількох завдань, які виконуються на одному ядрі, можна застосувати політику витіснення Earliest Deadline First (EDF) [17], яка завжди призначає найвищий пріоритет завданням з найближчим поточним дедлайном, а витіснення EDF може досягти 100-відсоткового використання кожного активного ядра.

ВИСНОВКИ

Оскільки індустрія 4.0 вступила в еру багатоядерних і багатопроцесорних систем, енергоефективність стає більш важливим критерієм при розробці кінцевих пристроїв IoT та систем реального часу. У даній статті представлено нову техніку для планування розкладу задач реального часу з максимальною енергоефективністю при заданих обмеженнях на пікову температуру. Результати показують, що підхід теплового балансування призводить до значного покращення енергоефективності та доцільності розбиття задач, особливо коли задане температурне обмеження є жорстким або використання системи є високим.

Література:

1. F. Yao, A. Demers, S. Shenkerю A scheduling model for reduced CPU energy. in: *Proceedings of IEEE 36th Annual Foundations of Computer Science*, 1995, pp. 374-382, doi:10.1109/SFCS.1995.492493.
2. D. Li, J. Wu. Minimizing energy consumption for frame-based tasks on heterogeneous multiprocessor platforms. *IEEE Transactions on Parallel and Distributed Systems*. 26 (3) (2015) 810-823, doi:10.1109/TPDS.2014.2313338.

3. Fan, M., Rong, R., Liu, S. et al. Energy calculation for periodic multi-core scheduling in system thermal steady state with consideration of leakage and temperature dependency. *J Supercomput* 71, 2565-2584 (2015). URL: https://doi.org/10.1007/s11227-015-1405-0
4. Benedikt, Ondřej, et al. Thermal-aware scheduling for MPSOC in the avionics domain: Tooling and initial results. *2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications* (RTCSA). IEEE, 2021
5. Nath, Shubha Brata, et al. A survey of fog computing and communication: current researches and future directions. URL: arXiv preprint arXiv:1804.04365 (2018)
6. Li C. et al. Edge-oriented computing paradigms: A survey on architecture design and system management. *ACM Computing Surveys* (CSUR). – 2018. – T. 51. – №. 2. – C. 1-34.
7. Chen, Yen-Kuang, and Ishmael F. Santos. Apparatus and method for reducing power consumption on simultaneous multi-threading systems. U.S. Patent No. 7,653,906. 26 Jan. 2010.
8. Deng, Zexi, et al. Energy-aware task scheduling on heterogeneous computing systems with time constraint. *IEEE Access*, 2020, 8: 23936-23950.
9. Almalki F. A. et al. Green IoT for eco-friendly and sustainable smart cities: future directions and opportunities. *Mobile Networks and Applications*. – 2021. – P. 1-25.
10. Zahaf H. E. et al. Energy-efficient scheduling for moldable real-time tasks on heterogeneous computing platforms. *Journal of Systems Architecture*. – 2017. – 74. – P. 46-60.
11. Hassan H. A., Salem S. A., Saad E. M. Energy aware scheduling for real-time multi-core systems. *International Journal of Computer Science Engineering* IJCSE. – 2018. – 7. – №. 4. – P. 167-175.
12. Saifullah A. et al. CPU energy-aware parallel real-time scheduling. *Leibniz international proceedings in informatics*. – 2020.
13. Zhang, Huazhe, and Henry Hoffmann. PoDD: Power-capping dependent distributed applications. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 2019.
14. Liu, Haoran, Jiaqi Yu, and Ruzhu Wang. Model predictive control of portable electronic devices under skin temperature constraints. *Energy* 260 (2022): 125185.
15. Rupanetti D., Salamy H. Thermal and energy-aware utilisation management on MPSoC architectures. *International Journal of Parallel, Emergent and Distributed Systems*. – 2021. – 36. – №. 5. – P. 449-469.
16. Hajkazemi, Mohammad Hossein, Mohammad Khavari Tavana, and Houman Homayoun. Wide I/O or LPDDR? Exploration and analysis of performance, power and temperature trade-offs of emerging DRAM technologies in embedded MPSoCs. *2015 33rd IEEE International Conference on Computer Design* (ICCD). IEEE, 2015.
17. Javed, Farhana, et al. Internet of Things (IoT) operating systems support, networking technologies, applications, and challenges: A comparative review. *IEEE Communications Surveys & Tutorials* 20.3 (2018): P.2062-2100.

МАТЕРІАЛИ

IV ВСЕУКРАЇНСЬКОЇ СТУДЕНТСЬКОЇ НАУКОВОЇ КОНФЕРЕНЦІЇ

17 ЛИСТОПАДА 2023 РІК • М. ЛЬВІВ, УКРАЇНА

РОЗВИТОК СУЧАСНОЇ НАУКИ: АКТУАЛЬНІ ПИТАННЯ ТЕОРІЇ ТА ПРАКТИКИ

ISBN 978-617-8126-72-8
DOI 10.36074/liga-ukr-17.11.2023

Вінниця, Україна
«UKRLOGOS Group»
2023

УДК 082:001
Р 64

Голова оргкомітету: Коренюк І.О.
Верстка: Зрада С.І.
Дизайн: Бондаренко І.В.

Конференцію зареєстровано Державною науковою установою «УкрІНТЕІ» в базі даних науково-технічних заходів України та інформаційному бюлетені «План проведення наукових, науково-технічних заходів в Україні» (Посвідчення №327 від 16.06.2023).

Р 64 **Розвиток сучасної науки: актуальні питання теорії та практики**: матеріали IV Всеукраїнської студентської наукової конференції, м. Львів, 17 листопада, 2023 рік / ГО «Молодіжна наукова ліга». — Вінниця: ТОВ «УКРЛОГОС Груп», 2023. — 526 с.

ISBN 978-617-8126-72-8
DOI 10.36074/liga-ukr-17.11.2023

Викладено матеріали учасників IV Всеукраїнської мультидисциплінарної студентської наукової конференції «Розвиток сучасної науки: актуальні питання теорії та практики», яка відбулася 17 листопада 2023 року у місті Львів, Україна.

УДК 082:001

ISBN 978-617-8126-72-8

Вишнєвський Дмитро Сергійович, магістр спеціальності 122 "Комп'ютерні науки"
Західноукраїнський національний університет, Україна

Науковий керівник: Осолінський Олександр Романович, канд. техн. наук, доцент кафедри інформаційно-обчислювальних систем і управління
Західноукраїнський національний університет, Україна

ПЛАНУВАННЯ ЗАВДАНЬ З УРАХУВАННЯМ ЕНЕРГОСПОЖИВАННЯ ДЛЯ ПРИСТРОЇВ ІНТЕРНЕТУ РЕЧЕЙ

Вступ

Інтернет речей (IoT), де десятки мільярдів взаємопов'язаних пристроїв спілкуються і співпрацюють один з одним через Інтернет, сьогодні привертає все більше уваги. Це пов'язано з багатьма причинами, але найважливішими з них є те, що ці пристрої спрямовані на підтримку та покращення повсякденного життя, вони дешеві та прості у використанні. Здебільшого ці пристрої оснащені акумулятором, радіочіпом та мікроконтролером (MCU) і одним або декількома датчиками та/або виконавчими механізмами.

З розвитком малопотужної та мініатюрної електроніки, а також радіотехнологій спостерігається явне зростання кількості застосувань Інтернету речей, що охоплюють широкий спектр областей застосування [1], таких як домашня автоматизація, носимі пристрої та промисловий або сільськогосподарський моніторинг.

IoT-пристрої часто страждають від піків струму через те, що більшу частину часу вони проводять у стані низького енергоспоживання (сплячому режимі).

Поєднання малої ємності накопичувачів енергії пристрої періодично вмикаються та вимикаються через часті перебої в електропостачанні рисунок 1). Це призводить до збою живлення.

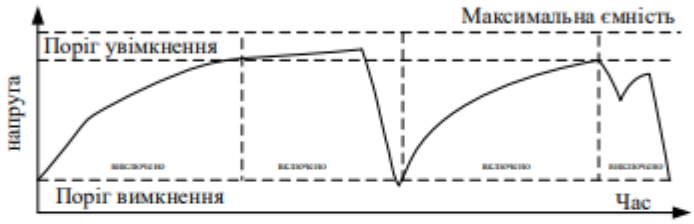


Рис. 1. **Переривчаста поведінка пристрою без батареї**

У даній статті акцент ставиться на обчислювальній частині, де представлено оптимальний алгоритм планування завдань з урахуванням енергоспоживання. Але для досягнення поставленої цілі програміст повинен провести декомпозицію додатку на набір взаємопов'язаних атомарних задач.

Запропонований підхід дозволяє уникати збоїв живлення, покращуючи загальну

продуктивність планувальника, а запропонований алгоритм дає змогу передбачити майбутній обсяг доступної енергії, необхідний для досягнення такого приросту.

Огляд відомих рішень

Традиційні моделі послідовних обчислень і мови програмування не можуть впоратися з такою різкою переривчастою поведінкою модулів IoT, оскільки вони передбачають безперервне виконання програмних інструкцій і покладаються на енергозалежну пам'ять. У роботі [2] досліджується існуючі схеми планування завдань для IoT, такі як динамічне масштабування напруги і частоти (DVFS), декомпозиція і комбінування завдань і робоче циклічне навантаження.

Існує дві основні обчислювальні стратегії - це моделі на основі контрольних точок та моделі на основі задач, де останні базуються на стратегії декомпозиції задач. В той час як моделі на основі контрольних точок, такі як в роботах [3] та [4], не є масштабованими через витрати часу та енергії на створення контрольних точок.

Як згадувалося раніше, ці моделі поділяють програму на різні атомарні підзадачі. Були запропоновані також інші підходи [5]. Однак витрати на апаратне забезпечення призводять до значного сповільнення роботи програмного забезпечення та збільшення складності.

У [6] було запропоновано алгоритм динамічного програмування для оптимізації планування завдань в IoT-пристроях щоб заздалегідь розрахувати оптимальне планування. У [7] автори представляють алгоритм адаптивної вибірки з урахуванням енергоспоживання.

Концепція планування завдань з урахуванням енергоспоживання

Як згадувалося в цій роботі пропонується система оптимізації, яка використовує два підходи: по-перше, розбиття завдання на атомарні підзадачі, по-друге, використання циклічного режиму роботи (рисунок 2).

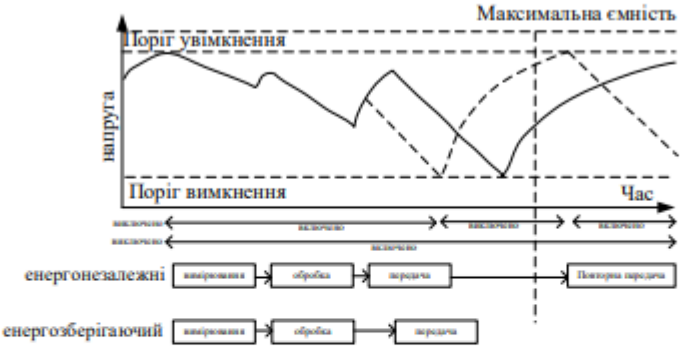


Рис. 2. **Планувальники завдань в безбатарейних пристроях, що не враховують енергоспоживання та враховують енергоспоживання**

Сучасні планувальники, часто не враховують енергію в своїх алгоритмах, в той час, як енергія джерел живлення в пристроях IoT є головною проблемою. На Рисунку представлено порівняння поведінки сучасного підходу, що враховує енергію.

У цьому прикладі додаток для вимірювання температури складається з трьох

завдань: вимірювання, обробка та передача. Однак, існуючі планувальники завдань для переривчастих обчислень вибирають наступне завдання для виконання на основі простих правил пріоритету, що призводить до потенційних тупиків.

Запропонований планувальник завдань, який враховує енергію, може визначати, яке завдання має бути виконане і коли, відповідно не тільки до трьох основних компонентів: доступної енергії в конденсаторі, енергії, що збирається, і енергоспоживання пристрою, що виконує завдання.

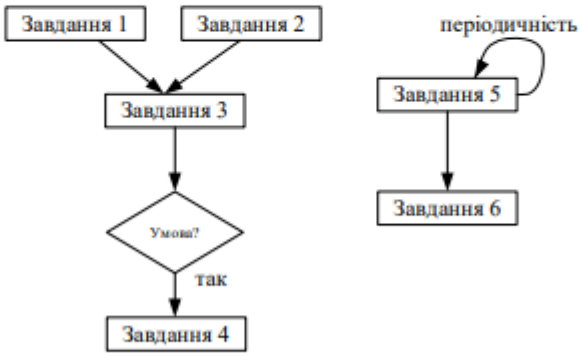


Рис. 3. **Загальний огляд моделі атомарних задач**

Основна мета енергоефективного планувальника - виконати максимальну кількість завдань, визначаючи їх пріоритетність.

Висновки

У даній статті було показано, що механізми планування з урахуванням енергоспоживання необхідні для покращення продуктивності успішного виконання додатків, що виконуються пристроями IoT. З цієї причини було представлено теоретичну інформацію про досяжний приріст продуктивності при плануванні завдань з урахуванням енергоспоживання порівняно з сучасними планувальниками завдань, які не враховують енергоспоживання.

Енергоефективний планувальник завдань дозволяє уникнути збоїв живлення, що дає змогу більшій кількості завдань виконатись без простою.

Список використаних джерел:

1. D. Ma, G. Lan, M. Hassan, W. Hu, and S. K. Das, "Sensing, computing, and communications for energy harvesting IoTs: A survey," IEEE Communications Surveys Tutorials, vol. 22, no. 2, pp. 1222–1250, 2020.
2. M. M. Sandhu, S. Khalifa, R. Jurdak and M. Portmann, "Task Scheduling for Energy-Harvesting-Based IoT: A Survey and Critical Analysis," in IEEE Internet of Things Journal, vol. 8, no. 18, pp. 13825-13848, 15 Sept.15, 2021, doi: 10.1109/JIOT.2021.3086186.
3. B. Ransford, J. Sorber, and K. Fu, "Mementos: System support for long-running computation on RFID-scale devices," in Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems, ser. ASPLOS XVI. New York, NY, USA: ACM, 2011, pp. 159–170. http://doi.acm.org/10.1145/1950365.1950386.

4. M. Hicks, "Clank: Architectural support for intermittent computation," in Proceedings of the 44th Annual International Symposium on Computer Architecture, ser. ISCA '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 228–240. https://doi.org/10.1145/3079856.3080238.
5. Delgado, Carmen, and Jeroen Famaey. "Optimal energy-aware task scheduling for batteryless IoT devices." IEEE Transactions on Emerging Topics in Computing 10.3 (2021): 1374-1387.
6. S. Escolar, A. Caruso, S. Chessa, X. d. Toro, F. J. Villanueva, and J. C. Lopez, "Statistical energy neutrality in IoT hybrid energy- 'harvesting networks," in 2018 IEEE Symposium on Computers and Communications (ISCC), 2018, pp. 00 444–00 449.
7. B. Srbinovski, M. Magno, F. Edwards-Murphy, V. Pakrashi, and E. Popovici, "An energy aware adaptive sampling algorithm for energy harvesting WSN with energy hungry sensors," Sensors (Switzerland), vol. 16, no. 4, pp. 1–19, 2016.