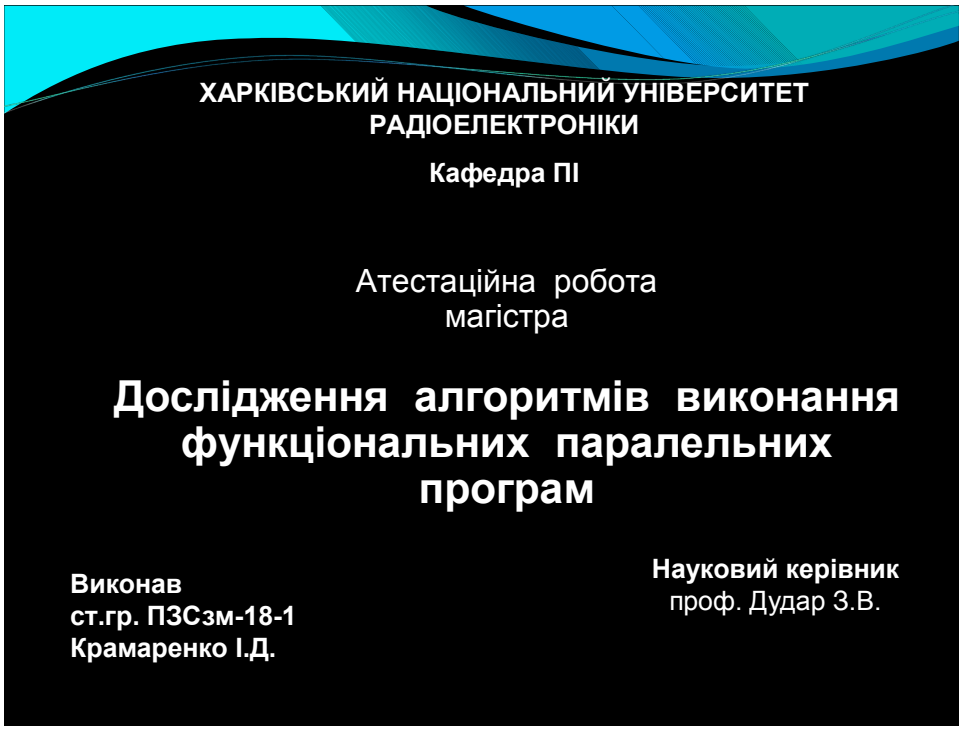
Додаток А

Програмні коди

```
scheme Integral
{
	Sqr = ([1]*[1]).mul;
	Integral = (0.0 * 2.0).Calc;

	fun Calc
	{
		Fs = ([1].Sqr * [2].Sqr).add;
		Dx = ([2] * [1]).sub;
		Trp = ((Fs * Dx).mul * 0.5).mul;
	}
}


scheme Functional
{
	Sqr = ([1]*[1]).mul;
	Functional = Trp(Sqr);

	fun Trp[F]
	{
		Fs = ([1].F * [2].F).add;
		Dx = ([2] * [1]).sub;
		Trp = ((Fs * Dx).mul * 0.5).mul;
	}
}


scheme FillArray
{
	FillArray = ([1] * ([1] * 0) . arrayCreate) . Fill; F =
	([2] * [1] * rand) . arrayAssign;

	fun Fill
	{
		N = [1];
		Arr = [2];
		Fill = (N * Arr * 0) . Recurse . [1];
		Recurse = ([3] * N) . equal -> Arr * Arr, (N * ([3] * Arr) .F * ([3]
* 1) . add) . Recurse;
	}
```

Додаток Б

Слайди презентації



1

Метою роботи є розробка й дослідження ефективності системи виконання функціональних паралельних програм мовою FPTL на багатоядерних комп'ютерах.

Для досягнення цієї мети в магістерській роботі вирішуються наступні завдання:

- порівняльний аналіз методів розпаралелювання обчислень у сучасних функціональних мовах програмування;
- дослідження методів і алгоритмів ефективного паралельного виконання функціональних FPTL програм на багатоядерних комп'ютерах;
- створення інтегрованої системи паралельного виконання FPTL програм на багатоядерних комп'ютерах.

2

Аналіз методів програмування багатоядерних комп'ютерів

- Одним з найпоширеніших методів програмування багатоядерних комп'ютерів є застосування **статичної багатопоточності** (static threading). Вона надає програмну абстракцію «віртуальних процесорів», або потоків (threads), що спільно використовують загальну пам'ять.
- Кожний потік підтримує пов'язаний з нею лічильник команд і може виконувати машинний код незалежно від інших потоків.

  *Одним з можливих рішень проблеми використання змінюваних даних у паралельнім середовищі є використання механізмів **статичного типового контролю**.*
- Рішенням проблеми організації конкурентного доступу до загальних даних є використання механізму програмної трансакційної пам'яті

3

Аналіз методів програмування багатоядерних компютерів

- Іншою проблемою статичної многопоточності є складність забезпечення **рівномірного розподілу роботи між потоками**.

- Для будь-яких (крім самих найпростіших) програм для збалансованого завантаження потоків програміст повинен розробляти складні протоколи взаємодії між потоками.
- Для вирішення проблем, що виникають при використанні статичних засобів розпаралелювання, була запропонована **модель динамічного розпаралелювання**. Що дозволяє програмісту вказувати рівень паралелізму в програмі, не турбуючись про комунікаційні протоколи, збалансованість завантаження та інші проблеми, які виникають при використанні паралельного програмування.

4

Приклад. Базисні функції для роботи з масивами

Ім'я функції	Вхідні дані		Вихідні дані		Опис
arraycreate	int	Довжина масиву.	Array['t]	Створений масив.	Створює масив заданої довжини й заповнює всі його елементи початковим значенням.
	't	Початкове значення.			
arrayassign	Array['t]	Масив.	Array['t]	Змінений масив.	Привласнює елементу масиву с заданим індексом задане значення.
	int	Індекс елемента.			
	't	Значення.			
arrayget	Array['t]	Масив.	't	Елемент масиву.	Повертає елемент масиву з заданим індексом.

5

Алгоритм типового контролю

- Як приклад розглянуто роботу алгоритму типового контролю на прикладі наступної програми:
- Нехай
- Робиться підстановка замість F2 терм т2 у правій частині рівняння
- Описано алгоритм, створений для ефективного виконання функціональних програм на багатоядерних комп'ютерах.

  В складі системи виконання функціональних програм можна виділити наступні підсистеми:

  - лексичний аналізатор – його призначення – перетворення вихідного тексту програми в список лексем;

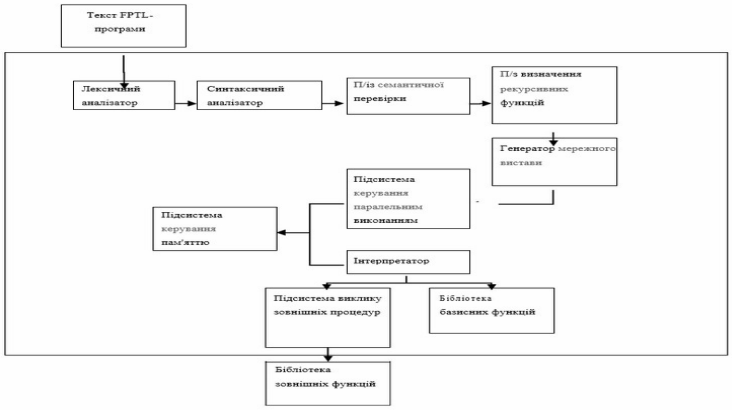
  - синтаксичний аналізатор - завдання цієї підсистеми – синтаксичний розбір списку лексем.

  У випадку відсутності в тексті програми синтаксичних помилок будується абстрактне синтаксичне дерево.

6

Підсистема керування пам'яттю

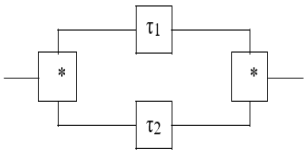
- відповідає за виділення й звільнення пам'яті при виконанні функціональної програми, робить автоматичне керування пам'яттю (прибирання сміття).
- На рівні програмної реалізації всі дані підсистеми об'єднані в один файл, що виконується. Архітектура системи виконання FPTL програм представлена на рис.



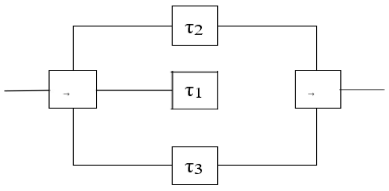
7

Структура системи виконання FPTL програм

- Представлення операції паралельної композиції



- Якщо $\tau = \tau_1 \rightarrow \tau_2, \tau_3$, то графічне подання має вигляд



8

- Завдання має наступну структуру:

```
Task (T, RS, DS,  ready).
```

```
EVALUATE (τ, T):
1  switch τ
2    case of BF(f, next):
3       f(T.DS)
4       EVALUATE(next, T)
5    case of FV(F, next):
6       PUSH(T.RS, next)
7       EVALUATE(τ_F, T)
8    case of FORK(τ_top, τ_bottom, next):
9       T_c := CREATE-TASK(T, τ_top)
10      EVALUATE(τ_bottom, T)
11      WAIT-TASK(T_c)
12      EVALUATE(next, T)
13   case of JOIN(next):
14      return
15   case of COND(τ_p, τ_t, τ_e):
16      EVALUATE(τ_p, T)
17      D := CHECK-COND(T.DS)
18      if D = true
19         EVALUATE(τ_t, T)
20      else EVALUATE (τ_e, T)
21   case of RET():
22      τ_next := POP(T.RS)
23      EVALUATE(τ_next, T)
```

9

Алгоритм роботи процедур CREATE-TASK і WAIT-TASK, які були використані раніше в процедурі EVALUATE

```
CREATE-TASK (T, T):
hch := Task(T, [],.T,   DS, [], false)
WQ-PUSH (PO, h)
return h
```

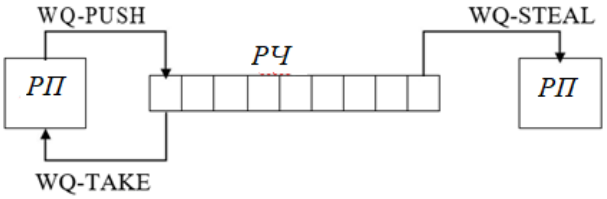
- Процедура **CREATE-TASK** створює новий *h*, який містить покажчик на вузол схеми й додає його в чергу РЧ робочого потоку .

```
WAIT-TASK ( ):
while ≠ true
SCHEDULE()
```

- Процедура WAIT-TASK робить очікування готовності породженого в процедурі CREATE-TASK завдання.
- Якщо завдання не готове, то відбувається виклик алгоритму планування (рядок 2) для пошуку іншої роботи. Таким чином, вдається уникнути простою потока. Процедура SCHEDULE реалізує алгоритм планування.

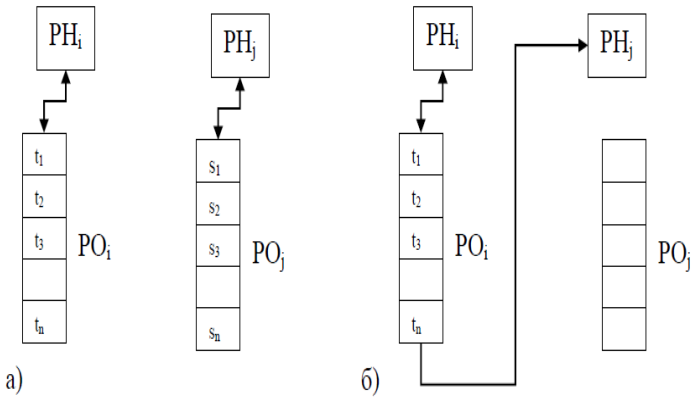
10

Організація взаємодії потоків із чергою завдань



11

Алгоритм роботи планувальника



Дії робочих потоків при:
а) наявності завдань у своїх чергах,
б) порожньої робочої черги в РП

12

Обчислення значень конструкторів і деструкторів

- Виклик зовнішніх функцій (foreign functions) реалізований через спеціальні процедури-перехідники.
- Для кожної зовнішньої функції створюється процедура-перехідник, яка витягає вхідні дані зі стека даних завдання, перетворює їх у формат вхідних параметрів процедури, робить виклик процедури, перетворює її вихідні параметри й поміщає їх назад у стек даних завдання.
- В описі вузлів мережного представлення під адресою зовнішньої функції розуміється адреса її процедури-перехідника.
- Усі використовувані в програмі бібліотеки зовнішніх функцій завантажуються в оперативну пам'ять безпосередньо перед виконанням FPTL програми

13

Програмні засоби, використані для реалізації компонентів системи виконання FPTL програм

Підсистема	Реалізація	Мова реалізації	Використовувані бібліотеки
Лексичний аналізатор	GNU Flex	C	
Синтаксичний	GNU Bison	C	
Семантичний	Власна	C++	
Підсистема пошуку рекурсивних рівнянь	Власна	C++	
Інтерпретатор	Власна	C++	
Робочі черги	Власна	C++	boost/atomic
Підсистема керування паралельним виконанням	Власна	C++	boost/thread
Складання сміття	Boehm GC	C	
Бібліотека базисних функцій	Власна	C++	
Підсистема виклику зовнішніх процедур	Власна	C	libffi

14

Висновки

Наведено аналіз системи виконання FPTL програм.

- описана організація взаємодії багатокомпонентної архітектури системи й підсистем.
- розроблене внутрішнє представлення функціональних програм у вигляді мереж ефективного виконання, що забезпечує FPTL програми на багатоядерних комп'ютерах, і представлення всіх структур даних, що використовуються у мові.

Розроблено алгоритми:

- планувальника паралельних процесів, що виконує розподіл завдань по робочих потоках;
- черг, що використовуються для ефективного зберігання породжених інтерпретатором завдань;

Розглянуто допоміжні аспекти організації роботи системи виконання FPTL-програм, а саме: автоматичне керування пам'яттю, представлення даних і виклики зовнішніх функцій.

Було зроблено **огляд** методів і програмних засобів, використаних для реалізації наведених алгоритмів і підсистем.

За результатами роботи подано до опублікування тези доповіді

Додаток В

Апробація результатів роботи

ДОСЛІДЖЕННЯ МОДЕЛЕЙ ПІДВИЩЕННЯ НАДІЙНОСТІ ФУНКЦІОНАЛЬНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Крамаренко І.Д.
Науковий керівник – проф. Дудар З.В.
Харківський національний університет радіоелектроніки
(61166, Харків, пр. Науки, 14, каф. Програмної інженерії,
тел.: (057) 702-14-46)
E-mail: andrii.dolzhykov@nure.ua

The proposed approach to software reliability evaluation allows using incomplete expert and statistical information about the functioning of the software system. The results obtained during the study create a theoretical basis for the development of algorithms aimed at increasing the efficiency of the processes of development and modernization of software systems. As an acceptable means of solving the problem of choosing a reliable software version, a genetic algorithm was chosen.

Найважливішим фактором, що визначає якість сучасного програмного продукту, є надійність його функціонування. Цією проблемою займаються дослідники й виробники ПЗ в усьому світі. Програмні збої, що приводять до зупинки виробництв або обслуговуванню клієнтів, помилки проектувальників, що приводять до витоків клієнтських даних – ці проблеми несуть колосальні фінансові й репутаційні ризики, як великим, так і малим компаніям. В той же час виробники програмного забезпечення змушені скорочувати строки розробки ПЗ, використовувати уніфіковані рішення для забезпечення сумісності програмних систем, залучати до розробки вилучених фахівців. Тобто незважаючи на високий запит на надійне ПЗ, розробники змушені заощаджувати.

Найефективніший спосіб забезпечити надійність функціонування програмного забезпечення – це проведення робіт на етапі проектування ПЗ. Вибір надійної архітектури програм дозволить уникнути більшості проблем і скоротити витрати на усунення проблем надійності в майбутньому. Однак у сучасних досліджених немає єдиного підходу до визначення терміна «надійність програмного забезпечення». Одні вкладають у це поняття несприйнятливість до зовнішніх впливів, інших – простоту й прозорість програмних рішень, що виключає можливість виникнення помилок.

Предметом дослідження є стохастичні алгоритми моделювання й вибору надійного варіанта програмного забезпечення, а об'єктом дослідження є стійка до відмов архітектура складних програмних систем.

Метою дослідження є підвищення обґрунтованості прийняття рішень при моделюванні процесу функціонування програмних систем, а також при виборі надійного варіанта програмного забезпечення.

Для досягнення поставленої мети вирішувалися наступні задачі: провести аналіз підходів оцінки надійності програмного забезпечення, модифікувати модель оцінки надійності функціонування програмної системи так, щоб вона дозволяла враховувати як експертні оцінки надійності окремих компонентів, так й оцінки, отримані в результаті статистичного аналізу експериментальних даних, модифікувати модель оцінки надійності функціонування програмної системи на основі оцінок надійності функціонування її компонентів, розробити стохастичні алгоритми багатокритеріальної умовної оптимізації, що дозволяють знаходити надійні варіанти архітектури ПЗ з обліком запрат на реалізацію ПЗ й можливістю використання різних підходів до забезпечення стійкості до збоїв, застосувати розроблені моделі й алгоритми при рішенні тестових і реальних практичних задач проектування архітектури ПЗ.

В ході виконання роботи запропоновано нову схему оцінювання рішень у багатокритеріальному генетичному алгоритмі, що відрізняється від відомих обліком одночасно всієї множини критеріїв і що дозволяє уникати передчасної збіжності алгоритму. Також розроблено спеціалізований генетичний алгоритм багатокритеріальної оптимізації, що дозволяє здійснювати пошук надійного варіанта програмної архітектури шляхом реалізації декількох варіантів підходу забезпечення надмірності програмних систем.

Основний результат магістерського дослідження складається в розробці спеціалізованих генетичних алгоритмів рішення задач багатокритеріальної оптимізації зі змінною довжиною хромосоми, що дозволяють одержувати в результаті своєї роботи репрезентативну апроксимацію множини Ф фронту Парето.

Розроблений алгоритм повинен працювати з варіантами ПЗ довільної конфігурації. Тобто, заздалегідь не відомо не тільки значення параметрів ПЗ, але і їхня кількість. Дійсно, якщо в якості одного з параметрів, що повинен настроїти генетичний алгоритм, виступає кількість компонентів, то серед варіантів ПЗ, які розглядає алгоритм, можуть виявитися такі, які відрізняються кількістю компонентів, а значить і кількістю наборів даних. Тому для рішення задачі вибору надійного варіанта ПЗ генетичний алгоритм повинен бути істотно модифікований, а саме створено генетичний алгоритм зі змінною довжиною хромосоми.

Запропонований підхід до оцінки надійності програмного забезпечення дозволяє використати неповну експертну й статистичну інформацію про функціонування програмної системи.

Результати, отримані при виконанні дослідження, створюють теоретичну основу для розробки алгоритмів, спрямованих на підвищення ефективності процесів розробки й модернізації програмних систем. В якості прийнятного засобу рішення задачі вибору надійного варіанта програмного забезпечення був обраний модифікований генетичний

Додаток Г

Відгук і рецензії

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ
Науково-навчальний центр заочної форми навчання

ВІГУК
на атестаційну роботу магістра

Крамаренка Івана Дмитровича, гр. ПЗСзм 18-1

спеціальність *121 – Інженерія програмного забезпечення*
освітньо-професійна програма - *«ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМ»*
Тема атестаційної роботи «Дослідження алгоритмів виконання функціональних паралельних алгоритмів»

Елементами наукової новизни та здійснення інновацій в умовах невизначеності вимог і ступінь складності розробки відповідає усім нормативним вимогам до магістерських робіт освітньо-професійної програми «програмне забезпечення систем», а саме досліджене та розроблено алгоритми підвищення ефективності системи виконання функціональних паралельних програм мовою FPTL на багатоядерних комп'ютерах. Існує багато систем-аналогів які в певній мірі могли б реалізувати поставлену задачу, та навіть вони повністю не задовольняють усім вимогам поставленим у цій роботі, тому було вирішено створювати власну програмну імітаційну систему.

В результаті роботи програма, якої відповідала більш висока надійність, мала більш просту структуру, і цей факт підтверджує гіпотезу, що встановилася, що ускладнення структури програм веде до зниження їх надійності.

Остаточний висновок щодо оригінальності роботи, враховуючи критерії оцінювання плагіату (п.4 Положення про протидію академічному плагіату в ХНУРЕ) – згідно звіту системи «Unicheck» ID перевірки:1000752673 – робота є повністю оригінальною.

Магістрант гр. ПЗСзм-18-1 Крамаренко Іван Дмитрович готовий до самостійної інженерної діяльності. Атестаційну роботу можна подати до захисту в ЕК за спеціальністю *121 – «Інженерія програмного забезпечення»*, освітньо-професійною програмою *Програмне забезпечення систем.*

Керівник атестаційної роботи магістра
З.В. Дудар, професор кафедри ПІ

UNICHECK

Власник документу:
Нечволод Вадим Юрійович каф. ПІ

ID перевірки:
1000752673

Дата перевірки:
10.12.2019 12:15:54 GMT+0

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
10.12.2019 14:20:52 GMT+0

ID користувача:
94949

Назва документу: 2019_ПІ_ПЗСзм-18-1_Крамаренко І.Д. _скорочений

ID файлу: 1000764419 Кількість сторінок: 38 Кількість слів: 7873 Кількість символів: 59842 Розмір файлу: 813.62 KB

0% Схожість

Не знайдено жодних джерел

0% Цитат

Не знайдено жодних цитат

0% Вилучень

Вилучений текст відсутній

Підміна символів

Заміна символів 19

Страница 1 из 39

Рецензія
на атестаційну роботу магістра
магістранта групи **ПЗСзм-18-1** ***Крамаренка Івана Дмитровича***
спеціальність – **121- Інженерія програмного забезпечення**
Освітня програма **Програмне забезпечення систем**

Тема атестаційної роботи:
«Дослідження алгоритмів виконання функціональних паралельних програм»

Структура атестаційної роботи: пояснювальна записка _____стор.; графічна частина _15_ аркушів; програмне застосування (прикладна програма) __6_ файла(ів) загальним обсягом _800_ Кбайт.

Застосування динамічних засобів розпаралелювання функціональних програм, заснованих на використанні паралельних платформ, проте, має свої недоліки. Написання коректних паралельних програм мовою Haskell з використанням операторів відносно легко, оскільки відсутність побічних ефектів означає, що програмісту не доводиться замислюватися про вирішення таких проблем, як стан перегонів або взаємоблокування, які можуть значно ускладнити написання й налагодження паралельних програм за допомогою засобів паралельного програмування. Відповідно до мети, ст. Крамаренко І.Д. виконав аналіз та удосконалення математичної моделі, аналіз можливих варіантів реалізації, проектування та створення методів та алгоритмів програмної системи.

Перевагами досліджуваних алгоритмів є: відсутність вимог до інформації про цільову функцію, здатність виходити з локальних оптимумів, низька ціна розробки й можливість інтеграції з іншими методами моделювання й оптимізації. В роботі створена програмна реалізація що ілюструє розроблені алгоритми.

Розглянуті джерела науково-технічної літератури, які слушно та в достатній мірі процитовані в тексті пояснювальної записки, та на їх основі зроблений висновок по розробці математичного апарату та програмного продукту. Пояснювальна записка оформлена грамотно, частини роботи збалансовані, добре представлена ілюстративна частина.

До недоліків можна віднести те, під час роботи програмного додатку є необхідність налаштування параметрів алгоритмів, зазначений недолік не впливає на загальну позитивну оцінку магістерської роботи.

Атестаційна робота магістранта гр. ПЗСзм-18-1 Крамаренка Івана Дмитровича відповідає вимогам до атестаційних робіт магістрів і заслуговує оцінки «добре (80)» і може бути представлена для захисту в ЕК.

Рецензент:
д.т.н., професор кафедри
програмної інженерії **І.Ю. Шубін**

Рецензія
на атестаційну роботу магістра
магістранта групи **ПЗСзм-18-1 Крамаренка Івана Дмитровича**
спеціальність – **121- Інженерія програмного забезпечення**
Освітньо-професійна програма **Програмне забезпечення систем**

Тема атестаційної роботи: «Дослідження алгоритмів виконання функціональних паралельних програм»

Структура атестаційної роботи:
пояснювальна записка _____стор.;
графічна частина _15__ аркушів;
програмне застосування (прикладна програма) _6__ файла(ів) загальним обсягом _800_ Кбайт.

Атестаційна робота Крамаренка І.Д. присвячена актуальній науково-технічній задачі застосування динамічних засобів розпаралелювання функціональних програм, заснованих на використанні паралельних платформ. Відповідно до мети, Крамаренко І.Д. виконав аналіз та удосконалення математичної моделі, аналіз можливих варіантів реалізації, проектування та створення методів та алгоритмів програмної системи.

В роботі створена програмна реалізація розроблених моделей.

Розглянуті джерела науково-технічної літератури, які слушно та в достатній мірі процитовані в тексті пояснювальної записки, та на їх основі зроблений висновок по розробці математичного апарату та програмного продукту.

Пояснювальна записка оформлена грамотно, частини роботи збалансовані, добре представлена ілюстративна частина.

До недоліків можна віднести те, під час роботі програмної системи не враховуються витрати оперативної пам'яті ПК, однак, зазначений недолік не впливає на загальну позитивну оцінку магістерської роботи.

Атестаційна робота магістранта гр. ПЗСзм-18-1 Крамаренка Івана Дмитровича відповідає вимогам до атестаційних робіт магістрів і заслуговує оцінки «добре – 80». Атестаційну роботу можна представити для захисту в ЕК.

РЕЦЕНЗЕНТ:

к.т.н., професор
каф. ІІІІ Н.В. Рябова