

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

Сергій СТИРЕНКО

(підпис)

“ ” _____ 2023 р.

Дипломний проект

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Комп’ютерні системи та мережі”

спеціальності 123 “Комп’ютерна інженерія”

на тему: Програмно-апаратний комплекс для автоматичної компенсації інструменту

Виконав : студент 4го курсу, групи ІО-з91
(шифр групи)

Крутько Олексій Михайлович

(прізвище, ім’я, по батькові)

(підпис)

Керівник доц., к.т.н., Корочкін О. В.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Консультант (нормоконтроль) ст. викладач Виноградов Ю.М.

(назва розділу)

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

(підпис)

Рецензент доц. каф. СПСКС к.т.н. Тарасенко-Клятченко О.В.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Засвідчую, що у цьому дипломному проекті
немає запозичень з праць інших авторів без
відповідних посилань.

Студент _____
(підпис)

Київ – 2023 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальність 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
Сергій СТИРЕНКО

(підпис)

“__” _____ 2023 р.

ЗАВДАННЯ

на бакалаврський дипломний проект студента

Крутько Олексія Михайловича

1. Тема проекту Розробка програмного комплексу для автоматичної компенсації інструменту

керівник проекту _____ доц., к.т.н, Корочкін О.В. _____,

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 31.05.2023 року № 2102-с

2. Термін здачі студентом закінченого проекту 20 травня 2023 р.

3. Вихідні дані до проекту технічна документація

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)
Опис предметної області, дослідження методики побудови компенсації інструменту

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень)
- | | |
|--|-------------|
| структурна схема ПАК | – Додаток А |
| діаграма класів ПАК(функціональна схема) | – Додаток Б |
| блок-схема алгоритму роботи потоків | – Додаток В |
| процес взаємодії Клієнта і Серверів (принципова схема) | – Додаток Г |
| процес взаємодії потоків в вузлах | – Додаток Д |
| текст програми | – Додаток Е |

6. Консультанти проекту, з вказівкою розділів проекту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 13.12.2022

Календарний план

№	Найменування етапів дипломного проекту	Терміни виконання етапів проекту	Примітки
1.	<i>Затвердження теми проекту</i>	<i>10.12.2022-15.12.2022</i>	
2.	<i>Вивчення та аналіз завдання</i>	<i>15.12.2022-15.03.2023</i>	
3.	<i>Розробка архітектури та загальної структури системи</i>	<i>15.03.2023-25.03.2023</i>	
4.	<i>Розробка структур окремих підсистем</i>	<i>25.03.2023-5.04.2023</i>	
5.	<i>Програмна реалізація системи</i>	<i>5.04.2023-15.04.2023</i>	
6.	<i>Оформлення пояснювальної записки</i>	<i>15.04.2023-04.06.2023</i>	
7.	<i>Захист програмного продукту</i>	<i>04.06.2023</i>	
8.	<i>Передзахист</i>	<i>05.06.2023</i>	
9.	<i>Захист</i>	<i>23.06.2023</i>	

Студент-дипломник

Крутько О. М.

(підпис)

(прізвище, ініціали)

Керівник проекту

Корочкін О. В.

(підпис)

(прізвище, ініціали)

Анотація

Даний бакалаврський проект присвячений створенню програмно-апаратного комплексу для вирішення задачі автоматичної компенсації інструменту при виготовленні стоматологічних протезів в умовах великого потоку замовлень в стоматологічній лабораторії. Останнім часом в стоматологічній індустрії стався масовий перехід на цифрові технології. Вхідні дані надходять до лабораторій не у вигляді гіпсових відбитків, а у вигляді 3д моделей. Зубні протези на основі цих моделей виготовляються на верстатах з числовим програмним керуванням. Для зубних протезів характерна умовно-довільна форма і вимагається достатньо велика точність виготовлення, для найкращого прилягання до препарованої поверхні зубів пацієнта. Оскільки інструмент, переважно свердло, верстата має кінцевий розмір, то необхідно виконати його компенсацію, інакше прилягання може бути незадовільним. Ця операція вимагає певних обчислювальних ресурсів.

Запропонований підхід полягає у створенні апаратного комплексу на основі сучасних комплектуючих, та програмної частини клієнт-сервер, які зможуть максимально ефективно виконувати компенсацію інструменту завдяки використанню паралельної та розподіленої обробки даних. Найбільший ефект у зменшенні часу виконання можуть отримати великі стоматологічні лабораторії, з кількістю робочих станцій більше десяти.

Annotation

The aim of this bachelor's project is to create a hardware and software system to solve the problem of automatic tool compensation while manufacturing dental prostheses in conditions of a large flow of orders in a dental laboratory. Recently, there has been a massive transition to digital technologies in the dental industry. The input data comes to the laboratories not gypsum impression, but rather as 3D models. Dental prostheses based on these models are manufactured CNC milling machines. Dental prostheses typically have an arbitrary shape and require high manufacturing accuracy for the best fit to the prepared surface of the patient's teeth. Since the tool, preferably a drill, has a finite size, it is necessary to compensate for it, otherwise the fit may be unsatisfactory. This operation requires significant computing resources.

The proposed method consists of creating a hardware complex based on existing components and a client-server software part that will be able to perform tool compensation as efficiently as possible due to the use of parallel and distributed data processing. Large dental laboratories with more than ten workstations can get the greatest effect in reducing the execution time.

ОПИС АЛЬБОМУ
ДО ДИПЛОМНОГО ПРОЕКТУ

На тему: «Програмно-апаратний комплекс для автоматичної компенсації
інструменту»

Київ 2023

[illegible]

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЕКТУ

На тему: «Програмно-апаратний комплекс для автоматичної компенсації
інструменту»

Київ 2023

ЗМІСТ

1	НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ.....	2
2	ПРИЧИНИ ДЛЯ РОЗРОБКИ	2
3	МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ.....	2
4	ДЖЕРЕЛА РОЗРОБКИ.....	2
5	ТЕХНІЧНІ ВИМОГИ.....	2
5.1	Вимоги до розробленого продукту	2
5.2	Вимоги до програмного забезпечення серверної частини.....	3
5.3	Вимоги до програмного забезпечення клієнтської частини.....	3
5.4	Вимоги до апаратного забезпечення серверної частини	3
6	ЕТАПИ РОЗРОБКИ	3

					ІАЛЦ.466500.002 ТЗ		
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>	Програмно-апаратний комплекс для автоматичної компенсації інструменту. Технічне завдання.		
Розроб.		Крутько О.М.					
Перевір.		Корочкін О.В.					
Н. Контр.		Виноградов Ю.М.					
Затверд.					КПІ ім. Ігоря Сікорського ФІОТ ІО-391		
					<i>Літ.</i>	<i>Арк.</i>	<i>Акрушів</i>
						1	3

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

Це технічне завдання поширюється на розробку програмно-апаратного комплексу для автоматичної компенсації інструменту. Область застосування: альтернатива теперішньому методу застосування компенсації інструменту в САПР в стоматологічних лабораторіях.

2 ПРИЧИНИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського проекту по освітньо-професійної програми “Комп’ютерні системи та мережі” спеціальності 123 “Комп’ютерна інженерія”, затверджене кафедрою Обчислювальної техніки Національного технічного Університету України “Київський Політехнічний інститут імені Ігоря Сікорського”.

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка програмно-апаратного комплексу для автоматичної компенсації інструменту.

4 ДЖЕРЕЛА РОЗРОБКИ

В якості джерел використовуються науково-технічна література із комп’ютерних технологій, паралельного програмування, наукові статті, публікації в мережі Internet.

5 ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до розробленого продукту

- Клієнт-серверна архітектура.
- Відмовостійкість – висока.
- Підтримка різних типів свердл

					ІАЛЦ.466500.002 ТЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		2

- Підтримка верстатів з різними ступенями свободи.

5.2 Вимоги до програмного забезпечення серверної частини

- Підтримка ОС MS Windows 10 і вище, Ubuntu 18.04.6 LTS і вище
- NET Framework: net6.0
- Засоби обміну даними по мережі: Socket або RPC

5.3 Вимоги до програмного забезпечення клієнтської частини

- Підтримка ОС MS Windows 10 і вище
- NET Framework: net6.0
- Засоби обміну даними по мережі: Socket або RPC

5.4 Вимоги до апаратного забезпечення серверної частини

- Можливість масштабування шляхом збільшення кількості вузлів.
- Топологія мережі з мінімально-необхідною кількістю з'єднань.
- Доступність комплектуючих.

6 ЕТАПИ РОЗРОБКИ

	Дата
6.1. Вивчення літератури	20.12.2022
6.2. Складання і узгодження технічного завдання	15.12.2022
6.3. Огляд і аналіз існуючих рішень	20.03.2023
6.4. Створення модулів програмно-апаратного комплексу	05.04.2023
6.5. Тестування у розподілений системі	15.04.2023
6.6. Налаштування і виправлення помилок	20.04.2023
6.7. Оформлення документації дипломного проекту	20.05.2023

					ІАЛЦ.466500.002 ТЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		3

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЕКТУ

На тему: «Програмно-апаратний комплекс для автоматичної компенсації
інструменту»

Київ 2023

ЗМІСТ

ВСТУП	4
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ.....	5
РОЗДІЛ 1 ПРОБЛЕМАТИКА ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ.....	6
1.1 Задача компенсації інструменту.....	6
1.2 Існуючі рішення	9
Висновки до розділу 1	10
РОЗДІЛ 2 ОГЛЯД АПАРАТНИХ І ПРОГРАМНИХ РІШЕНЬ ДЛЯ ПОБУДОВИ ВИСОКОПРОДУКТИВНИХ ПРОГРАМНО-АПАРАТНИХ КОМПЛЕКСІВ	12
2.1 Архітектури комп'ютерних систем.....	12
2.1.1 Класифікація архітектур КС	12
2.2 Центральні процесори	15
2.2.1 Процесори Intel™.....	15
2.2.2 Процесори AMD™.....	19
2.2.3 Порівняння процесорів.....	21
2.3 Оперативна пам'ять	23
2.4 Комп'ютерні мережі	23
2.5 Засоби паралельного програмування в платформі .NET	23
2.5.1 Бібліотека TPL.....	24
2.5.2 Розширення PLINQ.....	25
2.5.3 Планувальник задач.....	25
2.5.4 Потoki	25
2.5.5 Примітиви синхронізації.....	26
2.5.6 Структури даних для паралельного програмування	26
2.6 Організація розподілених обчислень.....	26
2.6.1 Сокети	27

					ІАЛЦ.466500.003 ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата	<i>Програмно-апаратний комплекс для автоматичної компенсації інструменту. Пояснювальна записка.</i>		
Розроб.		Крутько О.М.					
Перевір.		Корочкін О.В.					
Н. Контр.		Виноградов Ю.М.					
Затверд.							
					Літ.	Арк.	Акрушів
						1	58
					КПІ ім. Ігоря Сікорського ФІОТ ІО-391		

2.6.2 Віддалені процедури.....	28
Висновки до розділу 2	30
РОЗДІЛ 3 РОЗРОБКА АПАРАТНОЇ ТА ПРОГРАМНОЇ ЧАСТИНИ	
ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ	31
3.1 Конфігурація вузла	31
3.1.1 Центральний процесор	31
3.1.2 Системна плата.....	32
3.1.3 Оперативна пам'ять	32
3.1.4 Зберігання даних	33
3.1.5 Графічний адаптер	33
3.1.6 Мережевий адаптер	34
3.1.7 Система охолодження	35
3.2 Топологія комп'ютерної мережі.....	36
3.3 Розроблення паралельного алгоритму рішення задачі	36
3.4 Розроблення алгоритмів Серверів та Клієнта	39
3.5 Розроблення схеми взаємодії Серверів і Клієнта	40
3.6 Розроблення схеми взаємодії потоків в вузлах.....	41
3.7 Розроблення програми.....	41
3.7.1 ToolCompensation.....	42
3.7.2 Server	42
3.7.3 Client.....	43
3.7.4 ToolCompensationUI.....	43
Висновки до розділу 3	45
РОЗДІЛ 4 ТЕСТУВАННЯ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ ...	46
4.1 Тестування у розподіленій системі	46
4.2 Визначення ефективності програмно-апаратного комплексу.....	50
Висновки до розділу 4	54
ОСНОВНІ РЕЗУЛЬТАТИ І ВИСНОВКИ.....	55
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	56

ДОДАТОК А

ДОДАТОК Б

ДОДАТОК В

ДОДАТОК Г

ДОДАТОК Д

ДОДАТОК Е

ПІ. КОПІЇ ГРАФІЧНИХ МАТЕРІАЛІВ

					ІАЛЦ.466500.003 ПЗ	Арк.
						3
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

З розвитком цифрових технологій все більше галузей так чи інакше впроваджують цифрові рішення в свої підприємства. Не є винятком і стоматологічна галузь. Якщо ще два десятиліття тому стоматологічні клініки та лабораторії по всьому світу для виготовлення протезів використовували лише ручну роботу та відливання металевих конструкцій на основі гіпсових відбитків, то зараз багато з них використовують ручну роботу лише на етапах косметичного оброблення. Типовим процесом, наприклад, виготовлення зубної коронки в наші часи може бути наступне: пацієнт приходить в клініку, йому роблять відбиток зубів за допомогою інтраорального 3д-сканера, цей відбиток по мережі Internet пересилається в лабораторію, де за допомогою CAD системи моделюється коронка, яка по мережі Internet пересилається в центр з ЧПУ фрезувальними машинами, де коронка виготовляється і по пошті пересилається в клініку, де протягом наступного візиту пацієнта, вона йому вставляється. Очевидною перевагою даного підходу є відсутність ручної роботи, що високої точності роботи також дозволяє оптимізувати потоки виробництва.

Проте, як ми бачимо, цей процес пов'язаний з обробкою та пересиланням даних, а саме 3д-моделей. І оскільки вимагається точність порядку декількох десятків мікрон, 3д-моделі можуть досягати великих об'ємів, що ставить вимоги до швидкодії ПАК, який використовується.

Одним з важливих і вибагливих етапів у підготовці 3д-моделі, перш ніж її можна буде виготовити – є компенсація інструменту. Він дозволяє впевнитись, що кожна частина поверхні 3д-моделі може бути досягнута інструментом кінцевого розміру під час виготовлення. Переважно йдеться про свердло певного розміру. Тому я вважаю доцільним застосувати розподілені обчислення і спробую розробити ПАК для виконання компенсації інструменту у розподілений комп'ютерній системі.

					ІАЛЦ.466500.003 ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ПАК	Програмно-апаратний комплекс
PKCBA	Розподілена комп'ютерна система з багатоядерною архітектурою
SST	Speed select technology
IaaS	Logging as service
SaaS	Software as service
ШП	Центральний процесор
КС	Критична секція
ТС	Тип синхронізації
CAD	Computer-aided design
CAM	Computer-aided manufacturing
ЧУ	Числове управління
CISC	Complex instruction set computer
RISC	Reduced instruction set computer
MISC	Minimal instruction set computer
VLIW	Very long instruction word
SMP	Shared memory multiprocessing
MPP	Massive parallel processing

РОЗДІЛ 1 ПРОБЛЕМАТИКА ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Задача компенсації інструменту

Компенсація інструменту необхідна у багатьох галузях, де виготовлення деталей відбувається шляхом проходження інструменту на фрезерному верстаті і де поверхня деталі має щільно з'єднуватись з іншими частинами конструкції (Рис. 1.1) [1].



Рисунок 1.1 - Щільне з'єднання. Синій колір – деталь, що виготовляється, червоний – деталь, з якою виконується з'єднання.

Як можна побачити (Рис. 1.2), для того, щоб досягти найглибшої точки на внутрішній поверхні деталі, що виготовляється, інструмент певної форми та розміру має прибрати матеріал, який йому заважає.

Без додаткових дій, в такому випадку можливі два варіанти:

- 1 Алгоритм, який будує шлях інструменту, не дозволить інструменту змінювати поверхню, зміна якої не передбачалась. І тоді не відбудеться щільного з'єднання.
- 2 Алгоритм, який будує шлях інструменту, дозволить інструменту досягти найглибшої точки, але це призведе до зміни поверхні, що не

передбачалась. І тоді це може суттєво вплинути на характеристики деталі, наприклад, з точки зору супротиву матеріалу.

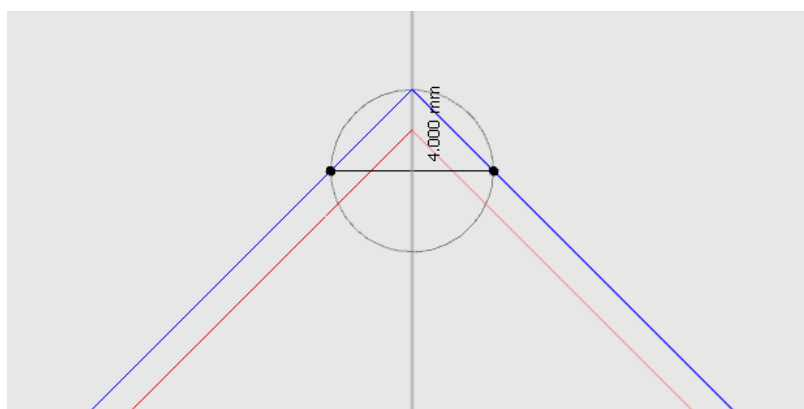


Рисунок 1.2 - Розташування інструменту сферичної форми для досягнення найглибшої точки на внутрішній поверхні деталі.

Тому в багатьох випадках важливо передбачити таку ситуацію і підготувати модель заздалегідь. А саме, самостійно змінити внутрішню поверхню так, щоб інструмент певного розміру зміг досягти кожної точки. І враховуючи це, забезпечити необхідні характеристики з точки зору супротиву матеріалу, як то, товщина, матеріал тощо.

Отже виходячи з параметрів інструменту, ступенів свободи верстату та форми моделі деталі, нам потрібно знайти всі можливі положення інструменту які:

- 1 Досягають кожної точки поверхні деталі
- 2 Вимагають зміни поверхні деталі.

Очевидно, що для випадків, коли деталь має примітивні форми, як, наприклад, див. рис. 1.1, всі такі положення інструменту можна визначити і задати аналітично. Це не є задачею, яка вимагає великих ресурсів для обчислення. Інша справа, якщо деталь має довільну форму. Однією з галузей, де деталі мають довільну форму, є цифрова стоматологія. В ролі деталей в цій галузі виступають протези, наприклад, коронки (Рис. 1.3), мости, шини.

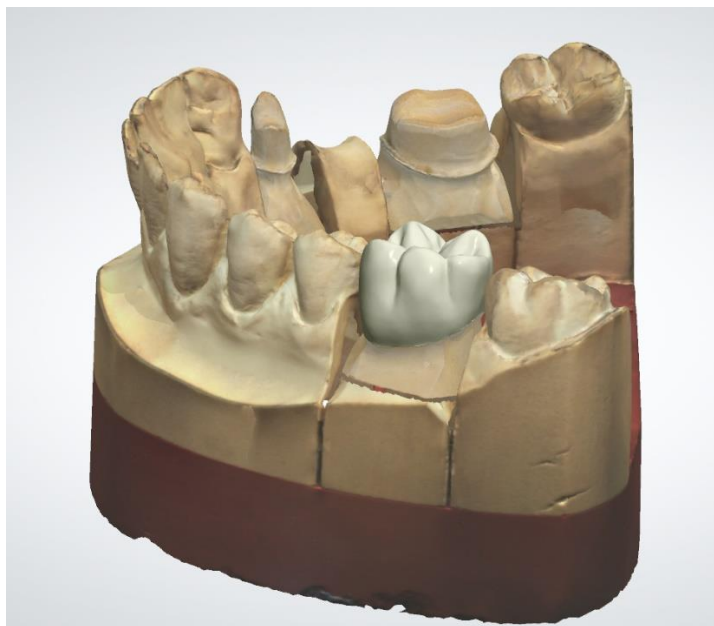


Рисунок 1.3 - Протез «коронка» на підготовленій поверхні. 3д модель.

На прикладі протезу типу «коронка» у розрізі (Рис. 1.4, 1.5) ми можемо побачити, що характер контактної поверхні абсолютно довільний і для того, щоб посадка такого протезу відбулась достатньо щільно, як того вимагає дизайн, потрібно врахувати спосіб, в який цей протез буде виготовлятись.

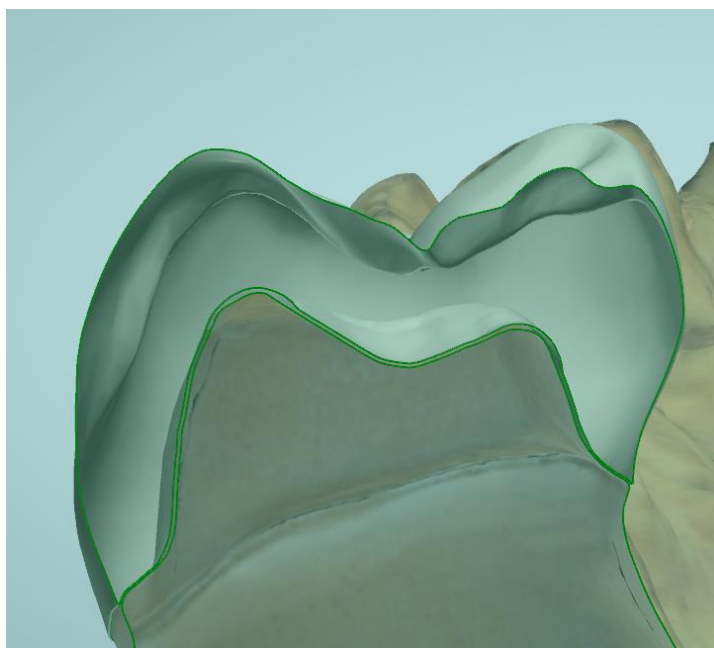


Рисунок 1.4 - Протез «коронка» у розрізі.



Рисунок 1.5 - Щільне прилягання протезу «коронка» до підготовленої поверхні.

Типовим матеріалом для коронок – є оксид цирконію, а метод виготовлення – фрезувальні верстати з числовим управлінням. Інструмент – свердло діаметром 1-2 мм. Отже якщо при моделювання такого протезу не врахувати компенсацію інструменту, то після виготовлення, протез або «сяде» на підготовлену поверхню недостатньо щільно, або досить крихкий матеріал не витримає навантажень вже після лікування пацієнта. В обох випадках це може мати дуже серйозні наслідки для пацієнта.

1.2 Існуючі рішення

Оскільки в проекті розглядається застосування компенсації інструменту при виготовленні стоматологічних протезів, зробимо огляд того, що пропонують лідери ринку в галузі цифрової стоматології. На 2023 рік найбільшими постачальниками програмних і апаратних продуктів для стоматологічних лабораторій є компанії Exocad [2] та 3Shape [3]. В CAD системах обох є підтримка компенсації інструменту [4, 5]. В якості параметрів оцінки існуючих рішень у таблиці 1.1 визначимо наступні параметри:

- форма інструменту – підтримка різних форм свердел

- розмір інструменту – підтримка різних розмірів свердел
- ступені свободи верстату – підтримка різних ступенів свободи верстатів з ЧУ
- паралельні обчислення – використання паралельних обчислень
 - віддалені обчислення – використання віддаленого серверу для обчислення важких операцій

Таблиця 1.1 – Характеристики існуючих рішень

Виробник	Форма інструменту	Розмір інструменту	Ступені свободи верстату	Паралельне обчислення	Віддалене обчислення
3Shape	Сферична	Діаметр	2 axis	Ні	Ні
Exocad	Сферична/Bullnose	Діаметр	2 axis	N/A	Ні

З результатів огляду існуючих рішень можна побачити, що провідні компанії не приділяли достатньо уваги паралельному обчисленню для виконання важких для обчислення типових операцій. Розподілене обчислення не використовується взагалі. Такий підхід викликає сумніви, особливо на тлі стрімкого розвинення хмарних сервісів, постійного зростання щільності ядер в центральних процесорах та зростанні швидкості передачі даних в комп'ютерних мережах. При досить високій вартості роботи дизайнера в стоматологічних лабораторіях – економія часу за рахунок прискорення виконання тривалих операцій – виглядає багатообіцяючою.

Висновки до розділу 1

Використання фрезерних верстатів для виготовлення стоматологічних протезів, де в свою чергу необхідна компенсація інструменту, залишається актуальним, навіть попри після суттєвого прориву тривимірного друку, оскільки залишаються актуальними матеріали, які поки не можливо друкувати.

Як можна побачити з огляду, лідери ринку не приділяють багато уваги застосуванню паралельних обчислень при обчисленні компенсації інструменту і задля того, щоб драматично не погіршувати користувацький досвід через наддовгий час виконання операції, обмежують точність обчислення та варіативність параметрів. Проте, аналіз користувацьких ресурсів, як то популярних форумів та каналів в YouTube, де професійні користувачі розповідають про свій досвід [6, 7], можна зробити висновок, що досить часто саме через недовготочність роботи компенсації інструменту виникають проблеми, коли виготовлені протези потребують мануальної доробки, або переробки.

Розробка ПАК для обчислення компенсації інструменту на віддаленому сервері потенційно дозволить збільшити точність обчислення, не збільшуючи мінімальні вимоги до апаратного забезпечення робочої станції. А також зменшить кількість випадків, коли виготовлений протез необхідно доробляти мануально.

РОЗДІЛ 2 ОГЛЯД АПАРАТНИХ І ПРОГРАМНИХ РІШЕНЬ ДЛЯ ПОБУДОВИ ВИСОКОПРОДУКТИВНИХ ПРОГРАМНО- АПАРАТНИХ КОМПЛЕКСІВ

2.1 Архітектури комп'ютерних систем

Головною ідеєю при проектуванні високопродуктивних обчислювальних систем є те, що задача вирішується на паралельних процесорах в p разів швидше, ніж на одному процесорі. Ця ідея могла бути реалізована в повному обсязі лише з появою потужної елементної бази, а саме з появою великих та надвеликих інтегральних схем та мікропроцесорів. Для побудови високопродуктивних паралельних обчислювальних систем використовують серійні або орієнтовані процесори [8].

Не менш важливою в багатопроцесорних системах є організація обміну даними між вузлами, використовуючи потужні мережі, та в межах вузла – використовуючи багаторозрядні швидкісні шини.

В цьому розділі буде зроблено огляд типів комп'ютерних систем, центральних процесорів, оперативної пам'яті та комп'ютерних мереж, на базі яких можна побудувати апаратну частину нашого комплексу.

2.1.1 Класифікація архітектур КС

Архітектури комп'ютерних систем класифікуються за наступними ознаками [9]

- Розрядність шин і машинних слів: 8, 16, 32, 64 і т.д. розрядні
- Система команд: CISC, RISC, VLIW, MISC
- Кількість процесорів: однопроцесорні, багатопроцесорні.
- Принцип взаємодії з пам'яттю: SMP, MPP, розподілені

Далі розглянемо деякі ознаки більш детально.

					ІАЛЦ.466500.003 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		12

Розрядність сучасних серійних ЦП в більшості випадків становить 64біт. Така довжина машинного слова дозволяє адресувати 128 терабайт адресного простору пам'яті, що на сьогоднішній день є більш ніж достатньо для всіх видів комп'ютерних систем, як низького рівня, так і суперкомп'ютерів. Для спеціальних інструкцій, призначених для прискорення тих чи інших операцій, і де є сенс у більшій розрядності, архітектуру ЦП розширюють спеціальним набором команд. Наприклад, MMX, SSE, AVX в архітектурі сімейства x86, компанії Intel.

CISC – комп'ютер зі складним набором інструкцій

Переваги

- Компактність набору інструкцій
- Менший розмір програм
- Підтримка програм високого рівня (менший семантичний розрив)

Недоліки

- Нерегулярність потоку команд
- Висока ціна апаратного забезпечення
- Складність з паралельністю

RISC – комп'ютер зі спрощеним набором команд

Переваги

- Нижча ціна апаратного забезпечення
- Нижче енергоспоживання

Недоліки

- На виконання ряду функцій потрібно більше команд
- Більший розмір програм
- Більше навантаження на пам'ять

VLIW – дуже довгі машинні команди

Переваги

- Можливість обробляти декілька інструкцій за такт при нижчій тактовій частоті

					ІАЛЦ.466500.003 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		13

- Проста архітектура процесора
- Нижче енергоспоживання

Недоліки

- Довжина програм може бути значно більше порівняно з суперскалярними архітектурами

MISC – комп'ютер з мінімальним набором інструкцій

Переваги

- Можливість обробляти декілька інструкцій за такт
- Компактна архітектура процесора, високі частоти
- Певне поєднання переваг RISC та VLIW

Недоліки

- Нижчий загальний паралелізм на рівні інструкцій

Принцип взаємодії з пам'яттю

SMP – симетрична архітектура, в якій два, або більше процесорів одним чином під'єднуються до спільної пам'яті, та виконують однакові функції.

Переваги

- Економічно найвигідніший спосіб масштабування обчислювальної системи – шляхом нарощування кількості процесорів(ядер)
- Обмін даними між процесорами через спільні змінні в пам'яті

Недоліки

- Вихід з ладу одного процесора призводить до некоректної роботи всієї системи. А вихід з ладу одного ядра в багатоядерному процесорі – виходу з ладу всього процесору.
- Навантаження на шину пам'яті при збільшенні кількості процесорів
- Проблема когерентності кешу

MPP – масивно-паралельна архітектура, в якій в кожного з наявних процесорів є свій блок пам'яті.

Переваги

- Висока масштабованість

					ІАЛЦ.466500.003 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		14

- Відсутність в необхідності в синхронізації при роботі с даними

Недоліки

- Менша швидкість міжпроцесорного обміну даними
- Вища складність максимального використання системних ресурсів

Беручі до уваги характер проблеми, яку ми збираємось вирішувати за допомогою ПАК, доступність комплектуючих потрібних для побудови апаратної частини, та баланс переваг і недоліків, доцільним виглядає обрати варіант архітектуру наступного типу: 64бітний багатоядерний процесор з системою команд CISC та SMP принципом взаємодії з пам'яттю.

2.2 Центральні процесори

При побудові апаратної частини обчислювального комплексу особливе значення має центральний процесор. Характеристики та тип ЦП суттєво обумовлює вибір інших складових, таких як пам'ять, прискорювачі, система охолодження, тощо. Серед компаній-виробників центральних процесорів найбільшими і найзначнішими за долею ринку на цей час є компанії Intel™ та AMD™. На базі їх багатоядерних процесорів побудовано безліч обчислювальних комплексів від рівня малого сервера до найпотужніших супер-комп'ютерів. Далі в цьому розділі буде зроблено огляд останнього на 2023 рік покоління процесорів від цих компаній, та порівняно їх характеристики, щоб при розробці нашого ПАК зробити вибір конкретного ЦП, а також оптимальних комплектуючих.

2.2.1 Процесори Intel™

Intel – найбільша в світі компанія виробник напівпровідникових елементів [10]. Зокрема відома завдяки своїм мікропроцесорам серії x86. Серед поточних лінійок процесорів нас цікавлять процесори побудовані на архітектурі amd64. Це є серії:

- Intel Core
- Intel Xeon

Обираючи лінійки процесорів для подальшого аналізу, відкинемо одразу ті, які очевидно не підходять для нашого ПАК. Наприклад, лінійки, які явною орієнтовані на персональних робочих станцій бюджетного рівня з префіксом i3, i5. Також нас не цікавлять серії з інтегрованим графічним ядром. А також обмежимося оглядом лише найновішою лінійкою на 2023й рік. Детальні характеристики у таблицях 2.1 та 2.2

Процесори серії Core – орієнтовані на середній рівень робочих станцій та серверів [11].

Таблиця 2.1 – Характеристики процесорів серії Core

	i9-13900F	i7-13700F	i9-13900KF	i7-13700KF
Дата випуску	Q1'23	Q1'23	Q4'22	Q4'22
Кількість ядер	24	16	24	16
Кількість потоків	32	24	32	24
Performance core Частота(базова), ГГц	2.0	2.1	3.0	3.4
Efficient Core Частота(базова) , ГГц	1.5	1.5	2.2	2.5
Performance core Частота(макс.) , ГГц	5.2	5.1	5.4	5.3
Efficient core Частота(макс.) , ГГц	4.2	4.1	4.3	4.2
Кеш, Мб	36	30	36	30

Кінець таблиці 2.1

	i9-13900F	i7-13700F	i9-13900KF	i7-13700KF
Контролерів пам'яті	2	2	2	2
Тип пам'яті	DDR4 – 3200 DDR5 – 5600	DDR4 – 3200 DDR5 – 5600	DDR4 – 3200 DDR5 – 5600	DDR4 – 3200 DDR5 – 5600
Макс. П.З. пам'яті, Гб/с	89.6	89.6	89.6	89.6
Сокет	FCLGA1700	FCLGA1700	FCLGA1700	FCLGA1700
Техпроцес, нм	7	7	7	7
TDP, Вт	65	65	125	125
Ціна, \$	524	359	564	384

Процесори серії Хеоп – орієнтовані для використання у неспоживчих робочих станціях, серверах та вбудованих системах. На відміну від серії Core підтримують більший об'єм пам'яті, більшу кількість каналів пам'яті, пам'ять з корекцією помилок та більшу масштабованість [12].

Поточне, на 2023й рік, 4те покоління серії має наступні підсерії:

- Н: Оптимізовано для баз даних та аналітики.
- М: Оптимізовано для ІІІ та обробки медіа.
- N: Оптимізовано для мереж.
- Р: Оптимізовано для IaaS хмар.
- Q: Підтримує рідкісне охолодження.
- S: Оптимізовано для систем зберігання даних.
- Т: Оптимізовано для тривалого використання.

- U: Обмеження одним сокетом.
- V: Оптимізовано для SaaS хмар.
- Y: Підтримка SST – технології контролю швидкості

Для нашого ПАК потенційно найкраще підходить серія Y, оскільки навантаження передбачається нерівномірно по часу.

Таблиця 2.2 – Характеристики процесорів серії Xeon

	6448Y	6444Y	6442Y	5418Y
Дата випуску	Q1'23	Q1'23	Q1'23	Q1'23
Кількість ядер	32	16	24	24
Кількість потоків	64	32	48	48
Частота(базова), ГГц	2.1	3.6	2.6	2.0
Частота(макс.), ГГц	4.1	4.0	4.0	3.8
Кеш, Мб	60	45	60	45
Контролерів пам'яті	8	8	8	8
Тип пам'яті	DDR5 - 4800	DDR5 - 4800	DDR5 - 4800	DDR5 - 4400
Сокет	FCLGA4677	FCLGA4677	FCLGA4677	FCLGA4677
Техпроцес, нм	7	7	7	7
TDP, Вт	225	270	225	185
Ціна, \$	3583	3622	2878	1483

2.2.2 Процесори AMD™

AMD – американська компанія, один із провідних виробників мікросхем, серед яких центральні процесори, графічні процесори, а також чіпсетів та багато іншого [13].

В сегменті центральних процесорів компанія AMD довгі роки є прямим і найбільш принциповим конкурентом Intel. Показово, що архітектура amd64, як нас цікавить в контексті побудови ПАК, є саме розробкою AMD. Серед лінійок актуальних процесорів AMD, аналогічно відкидаючи бюджетні версії та версії з інтегрованою графікою, слід дослідити наступні:

- Ryzen 7
- Ryzen 9
- Ryzen Threadripper
- EPYC

Детальні характеристики у таблицях 2.3 та 2.4.

Процесори серій Ryzen 7000 та Threadripper орієнтовані на середній рівень робочих станцій та серверів [14].

Таблиця 2.3 – Характеристики процесорів серії Ryzen

	9 7950X	9 7900X	7 7700X	Threadripper PRO 5975WX
Дата випуску	Q3'22	Q3'22	Q3'22	Q3'22
Кількість ядер	16	12	8	32
Кількість потоків	32	24	16	64
Частота(базова), ГГц	4.5	4.7	4.5	3.6

Кінець таблиці 2.3

	9 7950X	9 7900X	7 7700X	Threadripper PRO 5975WX
Частота(макс.), ГГц	5.7	5.6	5.4	4.5
Кеш L1, Мб	1	0.768	0.512	2
Кеш L2, Мб	16	12	8	16
Кеш L3, Мб	64	64	32	128
Контролерів пам'яті	2	2	2	8
Тип пам'яті	DDR5-3600 DDR5-5200	DDR5-3600 DDR5-5200	DDR5-3600 DDR5-5200	DDR4 - 3200
Сокет	AM5	AM5	AM5	sWRX8
Техпроцес, нм	5	5	5	7
TDP, Вт	170	170	105	280
Ціна, \$	550	450	400	3200

Процесори серії EPYC – орієнтовані для використання у неспоживчих робочих станціях, серверах та вбудованих системах. Кількість фізичних ядер варіюється від 8 до 32, а кількість логічних потоків від 16 до 64. В двох-процесорних системах кількість ядер може становити 64. На відміну від серії 7000 підтримують більший об'єм пам'яті, підтримують роботу з пам'яттю з корекцією помилок, підтримують більшу кількість каналів пам'яті та більшу масштабованість [15].

Таблиця 2.4 – Характеристики процесорів серії EPYC

	9254	9354P	9174F	9474F
Дата випуску	Q4'22	Q4'22	Q4'22	Q4'22
Кількість ядер	24	32	16	48
Кількість потоків	48	64	32	96
Частота(базова), ГГц	2.9	3.25	4.1	3.6
Частота(макс.), ГГц	3.9	3.75	4.15	3.95
Кеш L3, Мб	128	256	256	256
Контролерів пам'яті	12	12	12	12
Тип пам'яті	DDR5 - 4800	DDR5 - 4800	DDR5 – 4800	DDR5 - 4800
Сокет	SP5	SP5	SP5	SP5
Техпроцес, нм	5	5	5	5
TDP, Вт	200	280	320	360
Ціна, \$	2299	2730	3850	6780

2.2.3 Порівняння процесорів

Для порівняння представлених в огляді вище процесорів скористаємось дослідями одного з ведучих розробників програмного і апаратного забезпечення для аналізу та тестування швидкодії комплектуючих, компанії PassMark Software [16] у таблиці 2.5 та оцінимо співвідношення швидкодія/ціна у таблиці 2.6.

Таблиця 2.5 – Оцінка за синтетичним тестом PassMark

Процесор	Швидкодія
Intel XEON 6448Y	106832
AMD EPYC 9474F	99068
Intel XEON 5418Y	84673
AMD EPYC 9354P	79341
AMD Threadripper 5975WX	76120
AMD Ryzen 9 7950X	63581
Intel Core i9-13900KF	59646
Intel Core i9-13900F	53692
AMD Ryzen 9 7900X	52397
Intel Core i7-13700KF	46811
Intel Core i7-13700F	42447
AMD Ryzen 7 7700X	36547

Таблиця 2.6 – Оцінка співвідношення швидкодія/ціна

Процесор	Швидкодія/ціна
Intel Core i7-13700KF	122
Intel Core i7-13700F	118
AMD Ryzen 9 7900X	116
AMD Ryzen 9 7950X	115
Intel Core i9-13900KF	105
Intel Core i9-13900F	102
AMD Ryzen 7 7700X	91
Intel XEON 5418Y	57
Intel XEON 6448Y	30
AMD EPYC 9354P	29
AMD Threadripper 5975WX	23
AMD EPYC 9474F	15

Як можна побачити з таблиць 2.5 – 2.6, лідери за швидкістю поступаються більш бюджетним процесорам за співвідношенням швидкодія/ціна. Для оптимального вибору слід оцінити прогнозоване навантаження при цільовому використанні. Для розробки та тестування оптимальним виглядає саме процесор Intel Core i7-13700KF.

2.3 Оперативна пам'ять

Виходячи з огляду центральних процесорів, який ми зробили вище, можна констатувати, що всі процесори підтримують лише тип пам'яті DDR. Питання лежить лише в поколінні, пропускній здатності та наявності або відсутності підтримки контролю помилок. Характер обчислень не передбачає, що саме пам'ять може бути слабким місцем всього ПАК в цілому. Також, не є доцільним використання пам'яті з контролем помилок, тому що відмова звичайної пам'яті подія достатньо малоїмовірна, а шкода при цьому достатньо низька в нашому випадку. Тому питання вибору пам'яті швидше лежить в доступності комплектуючих. Для розробки і тестування цілком резонним виглядає тип пам'яті 32 Гб DDR4 – 3200 в двоканальному варіанті.

2.4 Комп'ютерні мережі

В якості комунікаційної мережі при побудові обчислювальних систем слід перш за все розглянути конкуруючі стандарти: Infiniband та група Ethernet.

2.5 Засоби паралельного програмування в платформі .NET

Платформа .NET це модульна платформа для розробки програмного забезпечення з відкритим кодом. Сумісна з операційними системами Windows, Linux та macOS. Вона активно розвивається та має власну спільноту на GitHub. На сьогоднішній день платформа забезпечує розширену підтримку паралельного програмування [26]. Пропонується середа виконання, бібліотечні класи та засоби відлагодження. Все це спрощує розробку паралельних програм а також дозволяє писати ефективний, деталізований та масштабований паралельний код. В платформі є все необхідне як для роботи з потоками і пулами потоків на низькому рівні, так і на високому завдяки бібліотеці паралельних задач (TPL). На Рис. 2.1 приведено загальний огляд архітектури паралельного програмування в .NET [17].

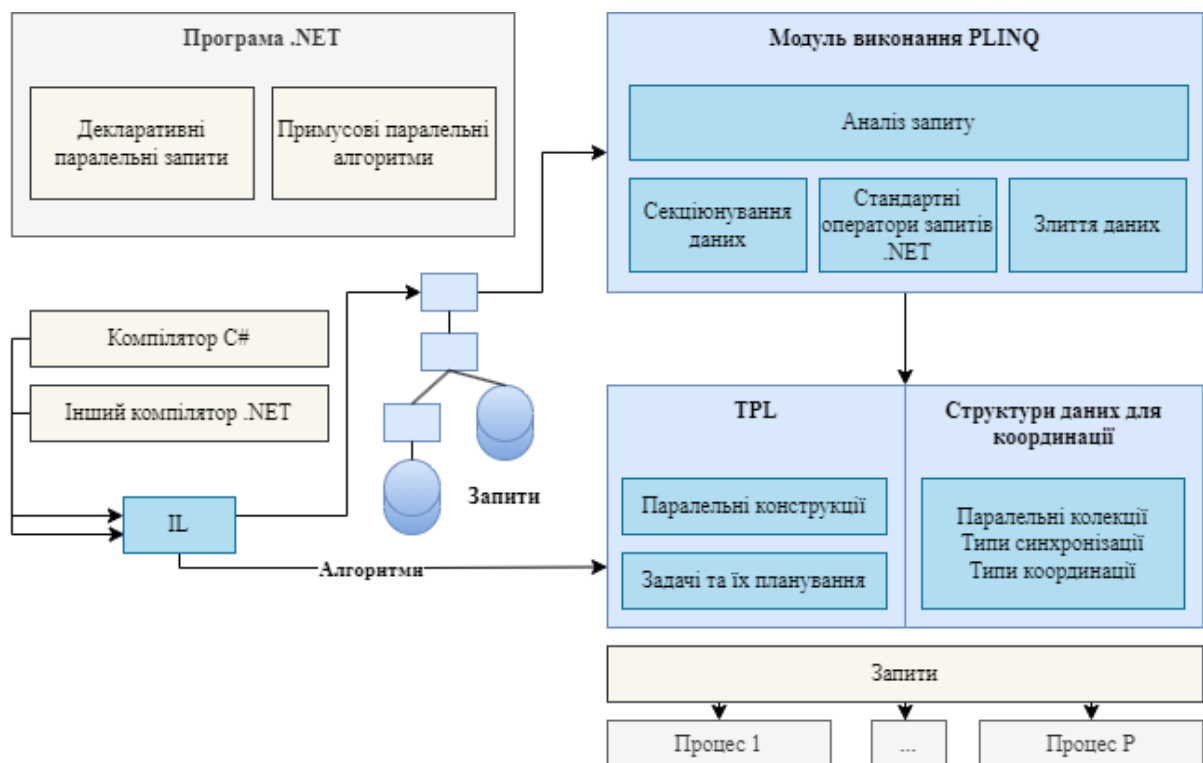


Рисунок 2.1 - Архітектура паралельного програмування в .NET

2.5.1 Бібліотека TPL

Task parallel library - бібліотека паралельних задач являє собою набір відкритих типів та API інтерфейсів доступних в просторі імен System.Threading та System.Threading.Tasks, головна ціль якої підвищення продуктивності праці при розробці паралельних програм. TPL здатна динамічно масштабувати рівень паралелізму та оптимізувати використання всіх наявних процесорів. Також підтримується розподілення роботи, планування задач, відміна виконання та інші низькорівневі засоби паралельного програмування [18].

Приклад циклу foreach в TPL:

```
// Послідовна версія
foreach (var item in sourceCollection)
{
    Process(item);
}

// Паралельна версія. Тіло циклу стає в вигляді лямбда виразу.
Parallel.ForEach(sourceCollection, item => Process(item));
```

2.5.2 Розширення PLINQ

Паралельне розширення LINQ – стандартної вбудованої мови запитів в C#. Поєднує простоту та зручність LINQ з потужністю паралельного програмування [19].

Приклад паралельного запиту:

```
var query = from item in
source.AsParallel().WithDegreeOfParallelism(2)
           where Compute(item) > 42
           select item;
```

2.5.3 Планувальник задач

Спеціальний об'єкт типу TaskScheduler який опрацьовує постановку задач в чергу на виконання. Можливо використовувати як версію за замовчуванням, так і реалізувати власну версію [20].

Приклад використання планувальника задач:

```
// Явне задання планувальника задач за замовчуванням.
Parallel.ForEach(sourceCollection,
new ParallelOptions { TaskScheduler = TaskScheduler.Current},
item => Process(item));
```

2.5.4 Потоки

Робота з потоками в .NET ведеться за допомогою класів Thread та ThreadPool [21].

При запуску процесу середа CLR автоматично створює один головний потік. Далі процес може створити інші.

Приклад створення і запуску потоку:

```
// Метод ThreadProc викликається, коли потік стартує.
// Чекає 1 секунду.
public static void ThreadProc() {
    Thread.Sleep(1000);
}

// Конструктор класу Thread вимагає ThreadStart
// делегат, як являє собою метод, що буде виконуватись в
// потоці.
var t = new Thread(new ThreadStart(ThreadProc));

// Запуск потоку та очікування на його завершення.
```

```
t.Start();
Thread.Sleep(0);
t.Join();
```

2.5.5 Примітиви синхронізації

Примітиви синхронізації в просторі імен System.Threading дають можливість детально налаштувати паралелізм. В наявності є як класичні примітиви синхронізації Event, Mutex, Semaphore, Monitor, Barrier та інші, так і багато їх варіацій для підвищення швидкодії в особливих випадках, наприклад: SemaphoreSlim який спрощеною версією Semaphore, яка не використовує системні функції Windows і є рекомендованою альтернативою для використання в межах одного додатка [22].

2.5.6 Структури даних для паралельного програмування

Платформа .NET включає в себе набір спеціальних структур, а саме колекцій для паралельного програмування [23]. Ці класи можна використовувати в будь якому паралельному коді, включно з PLINQ. Пропонуються наступні колекції:

- BlockingCollection<T>
- ConcurrentBag<T>
- ConcurrentDictionary<TKey, TValue>
- ConcurrentQueue<T>
- ConcurrentStack<T>

2.6 Організація розподілених обчислень

Організація розподілених обчислень так, чи інакше базується на обміні повідомленнями між вузлами та/або процесами. На платформі .NET реалізовано досить багато фреймворків за допомогою яких можна організувати розподілені обчислення. Серед найпоширеніших [24]:

- WCF – windows communication foundation – платформа для додатків орієнтованих на служби Windows.

- NET Remoting – API компанії Microsoft створений для організації міжпроцесорної взаємодії.
- NGrid – проект з відкритим вихідним кодом, для створення розподілених обчислень
- MPI.NET – реалізація стандарту MPI на C#

Також є можливість організувати обмін даними між задачами власноруч, використовуючи низькорівневі механізми, такі як сокети, або віддалені процедури.

2.6.1 Сокети

Механізм TCP сокетів в платформі .NET реалізований за допомогою класу Socket в просторі імен System.Net [25].

Приклад обміну повідомленнями:

```
// Створення сокету
using Socket socket = new Socket(SocketType.Stream,
ProtocolType.Tcp);
// Під'єднання до точки
socket.Connect(uri.Host, port);

// Відправлення запиту.
int bytesSent = 0;
while (bytesSent < requestBytes.Length)
{
    bytesSent+= socket
        .Send(requestBytes, bytesSent,
            requestBytes.Length - bytesSent,
            SocketFlags.None);
}

// Отримання відповіді
byte[] responseBytes = new byte[256];
char[] responseChars = new char[256];

while (true)
{
    int bytesReceived = socket.Receive(responseBytes);

    // 0 байт означає досягнення кінця повідомлення
    if (bytesReceived == 0) break;

    int charCount = Encoding.ASCII
```

```

        .GetChars(responseBytes, 0, bytesReceived,
responseChars, 0);
        // Друк отриманого повідомлення
        Console.Out.Write(responseChars, 0, charCount);
    }

```

2.6.2 Віддалені процедури

Для цього виду взаємодії реалізовано декілька бібліотек. Найбільш поширена на цей час крос-платформна бібліотека з відкритим вихідним кодом gRpc.

gRPC використовує підхід “contract-first”, тобто спочатку потрібно задати контракт – загальне визначення взаємодії [26]. Для цього потрібно зробити опис в форматі .proto

Приклад:

```

syntax = "proto3";
option csharp_namespace = "GreeterServiceApp";
package greet;

// Визначення сервісу
service Greeter {
    // Відправка повідомлення
    rpc SayHello (HelloRequest) returns (HelloReply);
}

// Повідомлення від клієнту містить name
message HelloRequest {
    string name = 1;
}

// Повідомлення клієнту містить message
message HelloReply {
    string message = 1;
}

```

Далі потрібно реалізувати сервіс.

Приклад реалізації сервісу:

```

using Grpc.Core;
using GreeterServiceApp;

namespace GreeterServiceApp.Services;

public class GreeterService : Greeter.GreeterBase
{

```

```

    public override Task<HelloReply> SayHello(HelloRequest
request, ServerCallContext context)
    {
        return Task.FromResult(new HelloReply
        {
            Message = "Hello " + request.Name
        });
    }
}

```

Таким чином, клієнт може викликати методи сервісу, який запущений віддалено.

					ІАЛЦ.466500.003 ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до розділу 2

В сучасному світі висока швидкість обчислень перш за все досягається збільшенням кількості процесорів, а не за рахунок складності архітектури ядра.

Для обчислювальних вузлів все ще найбільш актуальними є процесори з системою команд CISC, проте відомо, що на внутрішньому рівні CISC процесори перетворюють команди на RISC. Таким чином з одного боку зменшується семантичний розрив і знижується енергоспоживання.

AMD скоротила відставання від Intel в сегменті центральних процесорів в має конкурентноздатні продукти на всіх рівнях.

Відношення швидкодії/ціни не зростає для більш для процесорів з більшою кількістю ядер, а навпаки. Проте для серій Хеоп та ЕРҮС цей недолік компенсується завдяки високій масштабованості. В одному вузлі може бути розташовано до 12 процесорів.

Платформа .NET та зокрема мова С# пропонує розширені можливості для паралельного програмування та організації розподілених обчислень. Створено багато інструментів та бібліотек, які суттєво підвищують продуктивність розробника, проте, за потреби управління потоками та обмін повідомленнями між вузлами комплексу може бути реалізовано і на низькому рівні.

Розробка платформи .NET все більш рухається в напрямку відкритого коду, це з одного боку активно сприяє розвитку спільноти, але з другого боку слід бути уважним при виборі бібліотек, оскільки не всі вони підтримуються в повній мірі та своєчасно

РОЗДІЛ 3 РОЗРОБКА АПАРАТНОЇ ТА ПРОГРАМНОЇ ЧАСТИНИ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ

В даному розділі, з оглядом на результати аналізу проведеному в Розділі 2, виконано розробку апаратної та програмної частини ПАК, яка складається з багатоядерних вузлів з'єднаних у мережу. При виборі компонентів будемо керуватись сумісністю компонентів між собою, їх доступністю в Україні та ціною.

3.1 Конфігурація вузла

3.1.1 Центральний процесор

Центральний процесор грає ключову роль в системі, тому побудову слід почати саме з вибору ЦП, та обирати відповідні до нього інші компоненти.

16-ти ядерний процесор Intel Core i7-13700KF (рис. 3.1) є одним з найбільш оптимальних для даного ПАК.



Рисунок 3.1 - Intel Core i7-13700KF

3.1.2 Системна плата

Серед виробників системних плат, які підтримують процесор Intel Core i7-13700KF (Socket FCLGA1700) добре себе зарекомендувала компанія ASUS. Якісне виконання, підтримка DDR5, наявність слотів розширення, які необхідні для мережевого адаптера та відео адаптера, підтримка найшвидшого інтерфейсу для накопичувачів SSD – все це реалізовано в новій системній платі ASUS ROG MAXIMUS Z790 HERO (рис. 3.2).



Рисунок 3.2 – ASUS ROG MAXIMUS Z790 HERO

3.1.3 Оперативна пам'ять

Оперативна пам'ять Crucial (Рис. 3.3) відрізняється високою продуктивністю та енергоефективністю. Використовується як в настільних комп'ютерах так і в складних обчислювальних системах.

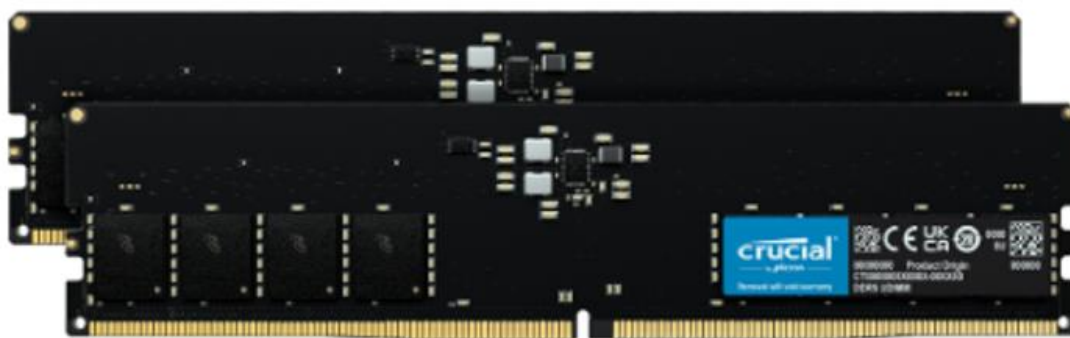


Рисунок 3.3 - Crucial 32GB Kit (2 x 16GB) DDR5-5600 UDIMM

3.1.4 Зберігання даних

Накопичувач в нашому ПАК – це на сам перед обслуговування операційної системи. Тому орієнтуватись слід на помірний об’єм, але максимальну швидкість. Модель Crucial P5 Plus (Рис. 3.4) цілком задовольнить потреби ПАК який розробляється.



Рисунок 3.4 - Crucial P5 Plus 1TB PCIe M.2 2280SS Gaming SSD

3.1.5 Графічний адаптер

Оскільки обчислення на які орієнтовано наш ПАК будуть виконуватись саме на ЦП, графічний адаптер не є слабким місцем, тому можна обмежитись одним з варіантів в нижньому сегменті графічних адаптерів. Також в цьому

випадку доцільно обрати варіант з пасивним охолодженням. Asus PCI-Eх GeForce GT 1030 (рис. 3.5) задовольняє вимогам.

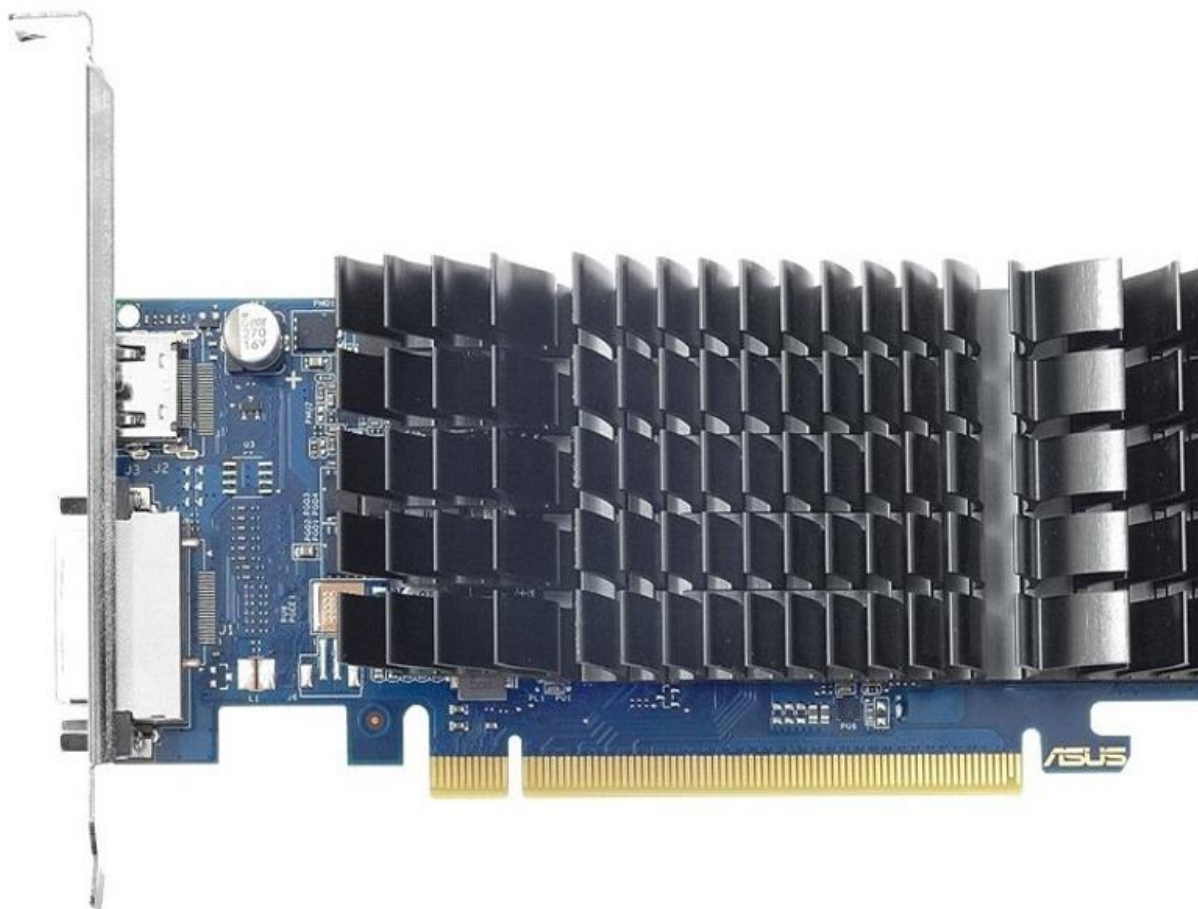


Рисунок 3.5 - Asus PCI-Eх GeForce GT 1030

3.1.6 Мережевий адаптер

Мережевий адаптер відіграє важливу роль в системах з розподіленими вузлами. В випадку нашої задачі об'єми даних, які пересилаються між вузлами можуть досягати десятків мегабайт за одну сесію. Проте частота появ сесій прогнозується невисокою. Мережевий адаптер Intel Ethernet Network Adapter X710-T2L (Рис. 3.6) цілком задовольнить потреби ПАК, що розробляється.



Рисунок 3.6 - Intel Ethernet Network Adapter X710-T2L

3.1.7 Система охолодження

Для розсіювання тепла, яке Intel Core i7-13700KF виробляє в піковому навантаженні становить 253 Вт потрібен відповідна повітряна, або рідкісна система охолодження. З міркувань вартості та важкості обслуговування, обмежимося повітряною системою. Deepcool AK620 (рис. 3.7) відповідає всім вимогам.



Рисунок 3.7 - Deepcool AK620

Таблиця 3.1 – Апаратна складова ПАК

Компонент	Назва	Ціна, грн
Процесор	Core i7-13700KF	17000
Системна плата	MAXIMUS Z790	27000
Оперативна пам'ять	32GB DDR5-5600	4500
Накопичувач	SSD 1TB	4000
Відео адаптер	GeForce GT 1030	3500
Мережевий адаптер	X710-T2L	15000
Система охолодження	Deercool AK620	2700
Сумарна вартість		73700

3.2 Топологія комп'ютерної мережі

В програмі Cisco Packet Tracer було розроблено і протестовано мінімальну конфігурацію (Рис. 3.8) ПАК з одною клієнтською робочою станцією та двома серверами. Маршрутизація забезпечується гігабітними маршрутизаторами.

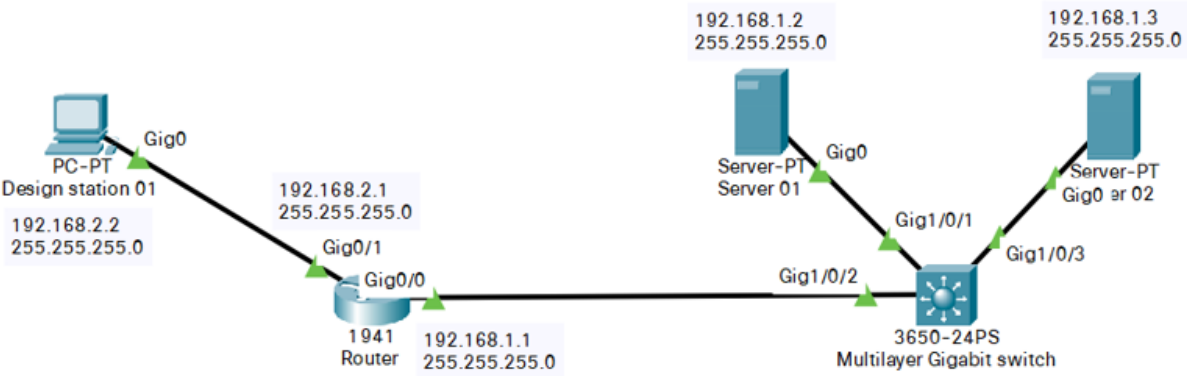


Рисунок 3.8 – Топологія мережі

3.3 Розроблення паралельного алгоритму рішення задачі

Структурна схема ПАК представлена у додатку А.

Вхідними даними для алгоритму є:

- Зд цілісна, «водонепроникна» поверхня, задана будь-яким чином: полігональна, полем відстаней, векторна, тощо. Модель орієнтована таким чином, що напрямок свердла співпадає з віссю Z. Розміри поверхні L- довжина, W-ширина, H-висота. (Рис. 3.9)
- Форма інструменту. Для спрощення припустимо, що інструмент це свердло зі сферичним кінцем. Параметрами інструменту тоді будуть радіус і довжина основи. R-радіус основи та сферичного кінця. Lb-довжина основи.
- Розмір сітки. Для кінцевого часу роботи алгоритму, потрібно квантувати поверхню, тому розіб'ємо її за допомогою сітки с кроком V (voxel size).

Задача алгоритму – мінімально змінити вхідну 3д поверхню так, щоб утворилась нова поверхня, де для кожної її точки існує дотична сфера, яка не перетинає цю поверхню.

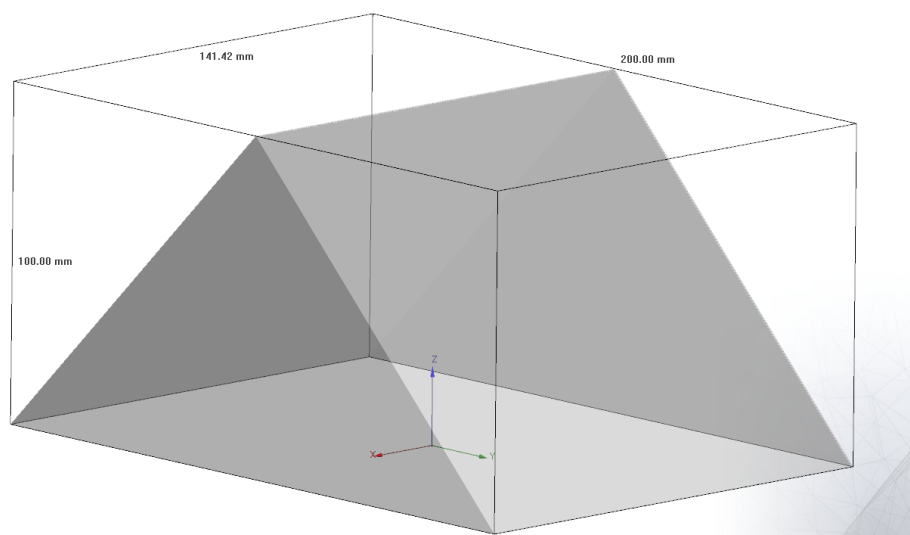


Рисунок 3.9 – Тестова 3д поверхня. Піраміда 200.0x141.43x100.0 мм.

Створимо двомірну мапу розміром

HeightMap[Hx, Hy], де $Hx = \text{Length}/\text{VoxelSize}$, $Hx = \text{Width}/\text{VoxelSize}$

Та заповнимо її значеннями компоненти Z відповідних точок вхідної поверхні. (Рис. 3.10)

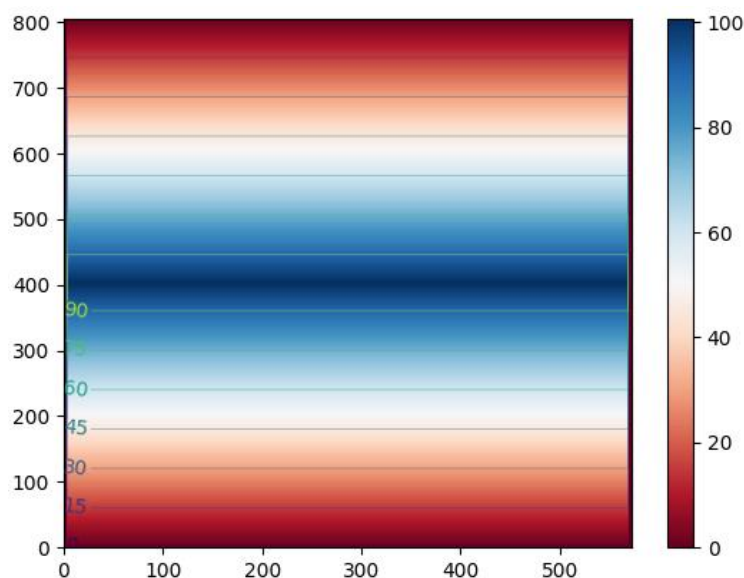


Рисунок 3.10 – Мапа висот 3д поверхні.

Для кожного елемента цієї мапи висот потрібно знайти таке положення інструменту, при якому точка лежить на поверхні сфери, а сума значень висот в точках, які потрапляють в сферу – мінімальна. Пошук позиції з мінімальною сумою можна виконати зміщуючи центральну позицію в усіх напрямках, та відслідковуючи зміну суми. Цей етап, очевидно, може бути виконаний паралельно. Випадки, коли мінімальна сума дорівнює 0, можна ігнорувати. Для прикладу, якщо вхідна поверхня є сферою з радіусом більшим, ніж радіус інструменту – алгоритм не має знайти жодної позиції з сумою більше 0.

Отримав всі позиції інструменту, можна адаптувати вхідну поверхню. Процес адаптації залежить від того, як саме задана поверхня. В багатьох випадках це складно виконати паралельно.

Паралельний алгоритм:

P – кількість процесорів

HeightMap – мапа висот, розміром H_x, H_y

$h = (H_x H_y) / P$ – розмір частини для обробки на одному процесорі

B – позиція інструменту

B_{opt} – оптимальна позиція інструменту

$Cost(B)$ – сума висот позиції

R – зсув центру сфери інструменту по X, Y, Z відносно поточної точки мапи висот.

1. $B[R, R, R]_h$ – всі можливі позиції із зсувом відносно точки.
2. $Cost(B[R, R, R]_h)_h$ – сума висот для кожного елемента $B[R, R, R]_h$
3. $Min(Cost(B[R, R, R]_h)_h)$ – мінімальна сума.
4. B_{opt}_h – позиції інструменту з мінімальною сумою для кожної точки

Спільним ресурсом буде колекція оптимальних позицій інструменту, доступ до якої потрібен задачам в серверній частині.

Блок-схема роботи алгоритму представлена у додатку В.

3.4 Розроблення алгоритмів Серверів та Клієнта

Кроки алгоритмів потоків сервера і клієнта з вказанням типів синхронізації та критичних секцій представлені в таблицях 3.2, 3.3, 3.4, 3.5, 3.6, 3.7

Таблиця 3.2 – Алгоритм обчислювального потоку (Сервер 1)

Крок	Дія	ТС, КС
1	Отримати від TS1 h_{2i}	W 1,1
2	Обчислити $B_{opt_h_{2i}}$	КС
3	Надіслати TS1 $B_{opt_h_{2i}}$	S 1,1

Таблиця 3.3 – Алгоритм потоку з'єднання з клієнтом (Сервер 1)

Крок	Дія	ТС, КС
1	Отримати від TC1 $Hm[H_x, H_y]_{h1}$, ToolRadius, VoxelSize	W c1,1
2	Надіслати TS2-N h_{1i}	S 2-N,1
3	Отримати від TS2-N $B_{opt_h_{1i}}$	W 2-N,1
4	Відіслати B_{opt_h1} TC1	S c1,1

Таблиця 3.4 – Алгоритм обчислювального потоку (Сервер 2)

Крок	Дія	ТС, КС
1	Отримати від TS1 h2i	W 1,1
2	Обчислити Bopt_h2i	КС
3	Надіслати TS1 Bopt_h2i	S 1,1

Таблиця 3.5 – Алгоритм потоку з'єднання з клієнтом (Сервер 2)

Крок	Дія	ТС, КС
1	Отримати від TC2 Nm[Hx,Hy]_h2, ToolRadius, VoxelSize	W c2,1
2	Надіслати TS2-N h2i	S 2-N,1
3	Отримати від TS2-N Bopt_h2i	W 2-N,1
4	Відіслати Bopt_h2 TC2	S c2,1

Таблиця 3.6 – TC1 Алгоритм потоку з'єднання з сервером 1 (Клієнт)

Крок	Дія	ТС, КС
1	Надіслати Nm[Hx,Hy]_h1, ToolRadius, VoxelSize TS1	S s1,1
2	Прийняти Bopt_h1 від TS1	W s1,1
3	Прийняти Bopt_h2 від TC2	W 1,1
4	Застосувати компенсацію інструменту	

Таблиця 3.7 – TC2 Алгоритм потоку з'єднання з сервером 2 (Клієнт)

Крок	Дія	ТС, КС
1	Надіслати Nm[Hx, Hy]_h2, ToolRadius, VoxelSize TS2	S s2,1
2	Прийняти Bopt_h2 від TS2	W s1,1
3	Надіслати Bopt_h2 в TC1	S 1,1

3.5 Розроблення схеми взаємодії Серверів і Клієнта

Для взаємодії клієнта та серверів створимо окремі спеціалізовані потоки, які будуть відповідати за ввід-вивід даних, створювати з'єднання та

пересилати дані між вузлами. ClientConnectionTask та ServerConnectionTask. Синхронізація буде відбуватись за допомогою примітиву Подія (ManualResetEvent). SendDoneEvent як сигнал, що відправлення повідомлення виконано та ReceiveDoneEvent як сигнал, що прийняття повідомлення виконано. Клієнт відправляє відповідну частину мапи висот, параметри інструменту та розмір сітки кожному серверу, та отримує відповідну частину оптимальних позицій інструменту.

Розроблена схема взаємодії серверів та клієнта представлена у додатку Г.

3.6 Розроблення схеми взаємодії потоків в вузлах

Кожен вузол, отримавши відповідну частину вхідних даних в ClientConnectionTask, розподіляє її між наявними робочими потоками CalculationTask. Для синхронізації використовується примітив Подія (ManualResetEvent) InputReadyEvent та масив ResultReadyEvents. Для захисту доступу до спільного ресурсу використовується неявний примітив Monitor реалізований в синхронізованій колекції ConcurrentBag.

Розроблена схема взаємодії потоків в вузлах представлена у додатку Д.

3.7 Розроблення програми

Розроблення пакету проектів ПАК проводилось в наступному середовищі та параметрами проекту:

- IDE: JetBrains Rider 2023.1.1
- .NET Core SDK: 7.0.302
- Platform: AnyCPU
- TargetFramework: net6
- Language version: C# 10.0

Пакет містить наступні проекти:

- ToolCompensation - бібліотека класів з реалізацією головного алгоритму

- Server - консольний додаток та реалізація серверної частини
- Client - бібліотека класів з реалізацією клієнтської частини
- ToolCompensationUI - допоміжний тестовий та демонстраційний додаток
- Benchmarks - допоміжний додаток для тестування швидкодії
- ToolCompensationTests - проект автоматичного тестування

Функціональна схема ПАК представлена у додатку Б

3.7.1 ToolCompensation

В даному проекті реалізовано центральний алгоритм ПАК, а саме інтерфейс ThreeZeroAxisToolCompensationBuilder. Конструктор класу вимагає лише об'єкт типу IToolShapeProvider, який представляє собою тип інструменту. Для спрощення, цей інтерфейс було реалізовано лише в SphericalTool, тобто інструмент в формі циліндру з напівкулею на одному з кінців. Для виконання алгоритму потрібно викликати метод BuildToolCompensation, передавши параметрами заздалегідь побудовану мапу висот для певної 3д моделі. На виході отримується масив позицій заданого інструменту для кожної точки мапи висот та їх вартість.

3.7.2 Server

В даному проекті реалізовано створення сокета на боці серверу, обмін повідомленнями(клас ClientConnectionSocket), потік взаємодії з клієнтом та потоки розрахунку(клас CalcServer), які викликають методи з бібліотеки класів ToolCompensation. Також реалізовано консольний додаток, який дозволяє працювати серверу в режимі сервісу, який обробляє запити клієнтів по мірі їх під'єднання (Рис. 3.11).

3.7.3 Client

В даному проекті реалізовано створення сокета на боці клієнту, обмін повідомленнями(клас `ServerConnectionSocket`), та потік взаємодії з сервером (клас `ClientCalc`). Розрахунок на боці клієнта безпосередньо в цьому проекті не ведеться – для цього є можливість запустити копію серверу на робочій станції клієнта.

3.7.4 ToolCompensationUI

Даний проект носить лише тестовий та демонстраційний характер. Також він собою являє достатнє наближення до систем автоматичного проектування, які використовуються в цифрових стоматологічних лабораторіях.

Інтерфейс користувача (Рис. 3.12) поділяється на три групи: панель управління, 3д відображення цільових та референтних моделей, та вікно для виводу лог-тексту. В конфігураційному файлі `servers.json` можна задати список серверів та їх параметри. В панелі управління можна завантажити потрібні моделі та задати параметри сітки та інструменту.

Вихідний код проектів `ToolCompensation`, `Server` та `Client` представлений у додатку Е.

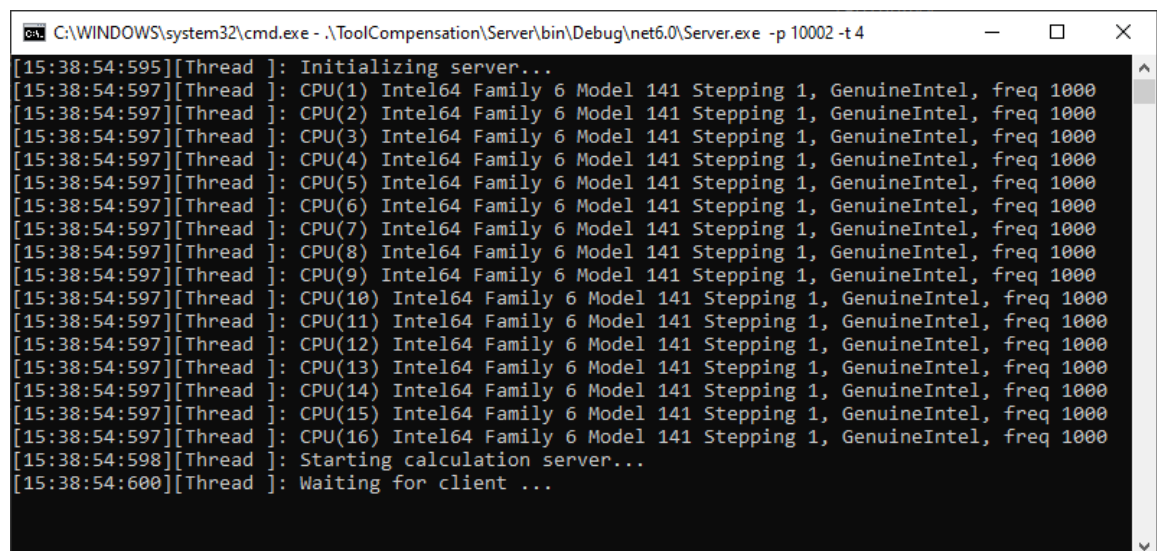


Рисунок 3.11 – Server в режимі очікування клієнта, консольний додаток.

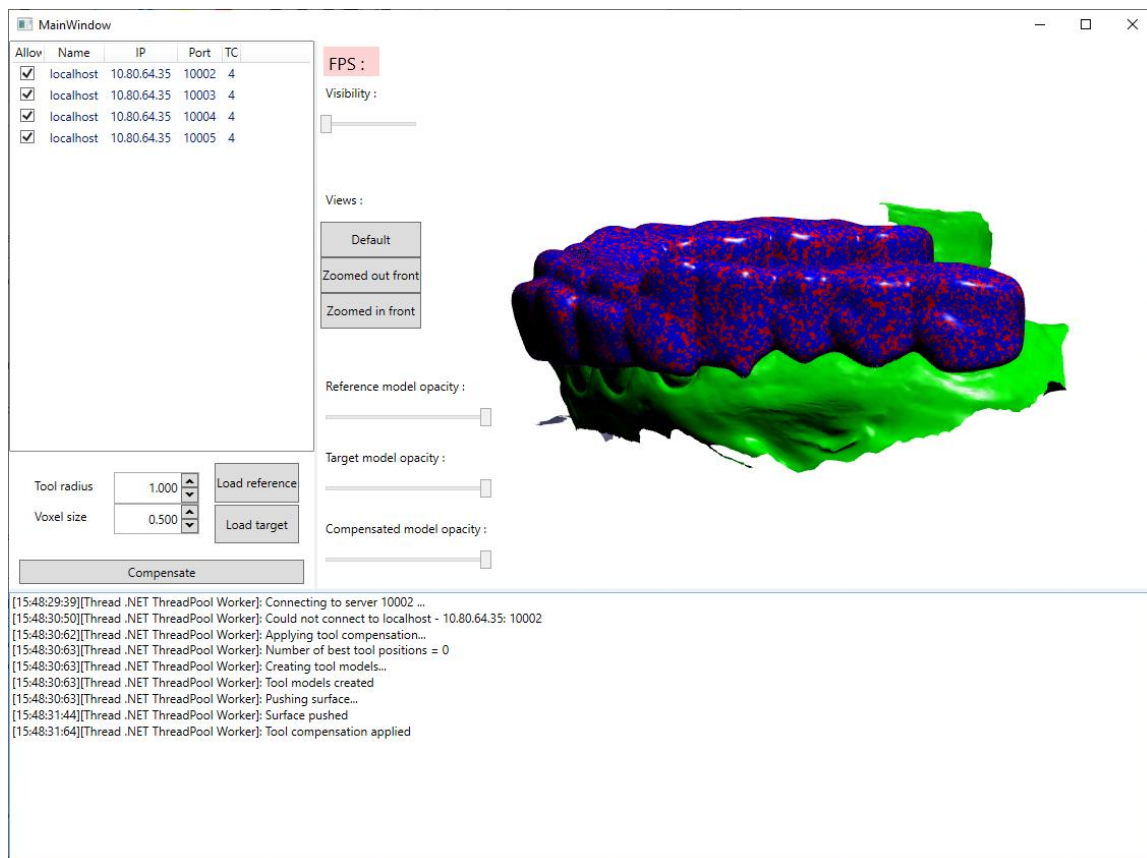


Рисунок 3.12 – Тестовий додаток.

Висновки до розділу 3

В даному розділі було обрано комплектуючі компоненти вузла ПАК. Головними критеріями при виборі стали: оптимальне співвідношення ціни-швидкодії, доступність на українському ринку та доступність сервісного обслуговування. Отримана конфігурація задовольнить невелику лабораторію, де до даного вузла будуть від'єднуватись до 10 клієнтських робочих станцій з системами САПР. Проте більш великі компанії можуть обрати конфігурацію яка задовольнить саме їх потреби, наприклад багатопроцесорні системи на базі процесорів Intel Xeon. Розроблена серверна частина ПАК може бути легко скомпільована для таких процесорів.

Також в даному розділі було описано та розроблено паралельний алгоритм рішення задачі компенсації інструменту та реалізовано клієнтську та серверну частини ПАК. Обрана мова програмування C#, та платформа dotNet цілком задовольнили наші потреби. Приємним додатком стала можливість, виконуючи головну розробку в ОС Windows, зібрати пакет серверної частини для ОС Linux.

В якості удосконалення даного ПАК, бажаною виглядає розробка планувальника для серверної частини. Задачею такого планувальника була б оптимізація використання ресурсів серверів відповідно до історії запитів від клієнтів. Поточна версія обов'язково зіштовхнеться зі складнощами, якщо навантаження по кількості запитів досягне критичного рівня.

РОЗДІЛ 4 ТЕСТУВАННЯ ПРОГРАМНО- АПАРАТНОГО КОМПЛЕКСУ

4.1 Тестування у розподіленій системі

Концептуальне тестування проведено в системі з наступними параметрами:

- CPU: 11th Gen Intel(R) Core (TM) i7-11850H @ 2.50GHz 2.50 GHz, 8 ядер, 16 потоків.
- RAM: 32 Gb DDR-3200
- SSD: 1 Tb
- Network: Intel Ethernet Connection I219-LM

Розподілену систему представимо як декілька копій серверу, кожна з яких використовує лише певну кількість ядер. Наприклад, 4 сервера по 2 ядра. Для цього створимо конфігураційний файл server.json

```
[
  {
    "Name": "localhost",
    "IP": "10.80.32.64",
    "Port": 10002,
    "ThreadsCount": 2,
    "IsAllowed": true
  },
  {
    "Name": "localhost",
    "IP": "10.80.32.64",
    "Port": 10003,
    "ThreadsCount": 2,
    "IsAllowed": true
  },
  {
    "Name": "localhost",
    "IP": "10.80.32.64",
    "Port": 10004,
    "ThreadsCount": 2,
    "IsAllowed": true
  },
  {
    "Name": "localhost",
    "IP": "10.80.32.64",
```

```
"Port": 10005,  
"ThreadsCount": 2,  
"IsAllowed": true
```

Та пакетний файл 4x2.bat

```
start cmd /k
.\ToolCompensation\Server\bin\Debug\net6.0\Server.exe -p 10002
-t 2
start cmd /k
.\ToolCompensation\Server\bin\Debug\net6.0\Server.exe -p 10003
-t 2
start cmd /k
.\ToolCompensation\Server\bin\Debug\net6.0\Server.exe -p 10004
-t 2
start cmd /k
.\ToolCompensation\Server\bin\Debug\net6.0\Server.exe -p 10005
-t 2
```

Сервери в режимі очікування клієнту (Рис. 4.1)

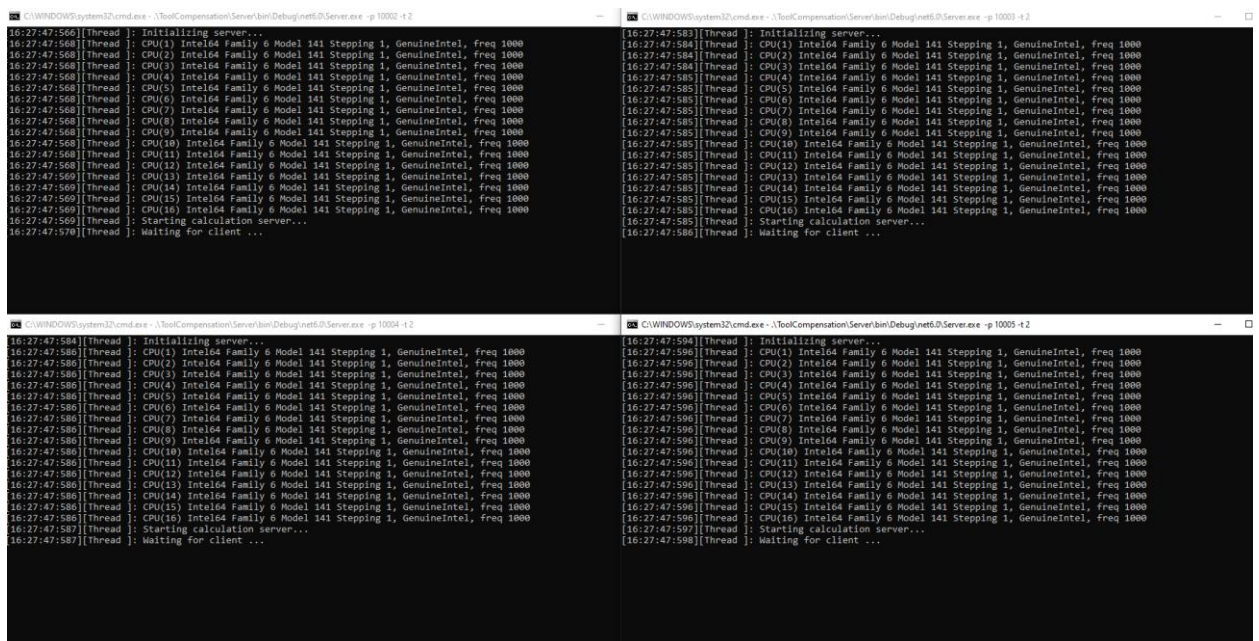


Рисунок 4.1. Сервери в режимі очікування

Запустимо клієнт та виконаємо операцію компенсації інструменту з розміром сітки 0.5 мм та радіусом інструменту 1.0 мм.

На Рис. 4.2 можемо бачити, що клієнт успішно під'єднався до всіх серверів, передав їм дані, отримав від серверів результат та застосував його до

синьої моделі. Це можна побачити на гранях піраміди. Розглянемо результат у розрізі (Рис. 4.3)

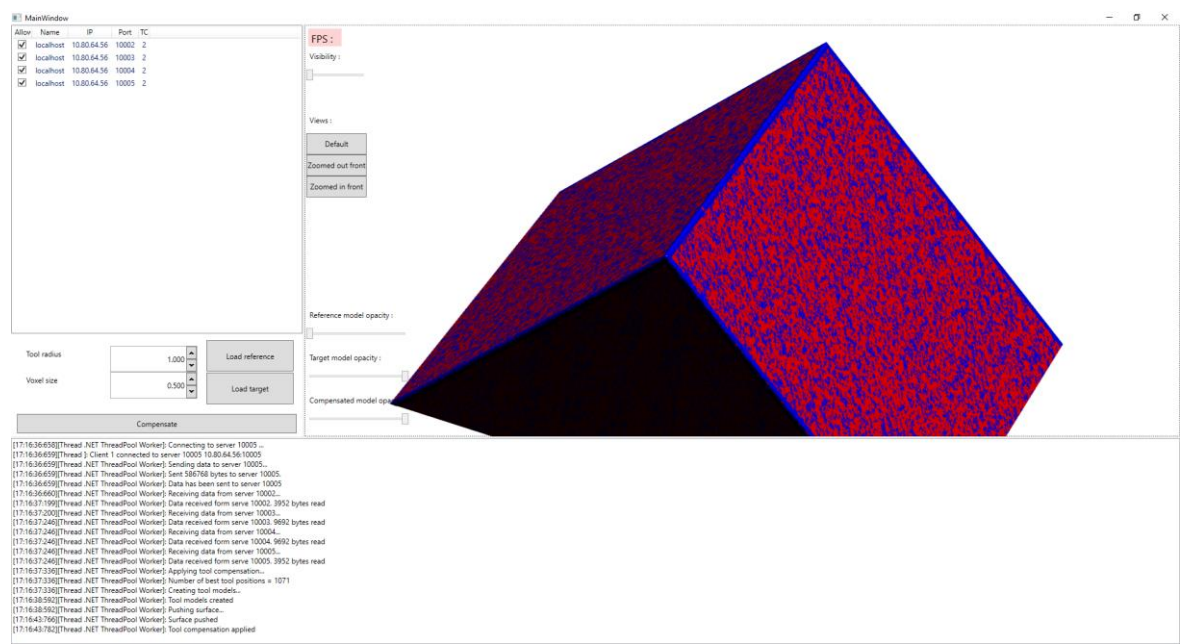


Рисунок 4.2 Операція компенсації виконана успішно

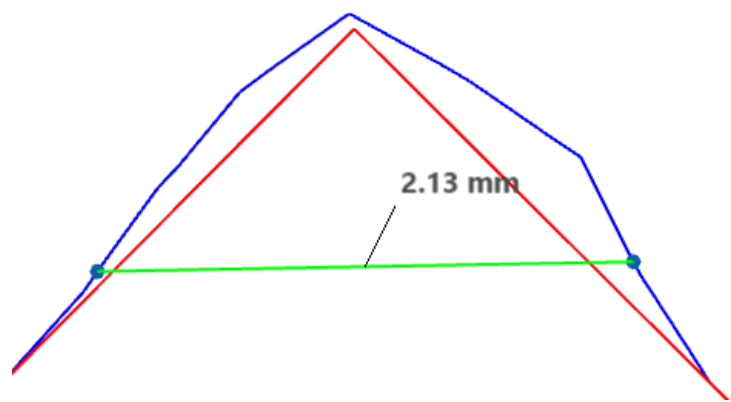


Рисунок 4.3 Результат компенсації інструменту. Розмір сітки 0.5мм, радіус інструменту 1.0мм

Як видно з результату роботи алгоритму, по-перше, синя поверхня далека від сферичної форми, по-друге, радіус кривизни суттєво відрізняється від очікуваного, а саме подвійного радіуса інструменту, тобто 2мм. Причиною тому стало саме обраний розмір сітки. Виконаємо операцію з розміром сітки 0.25 мм. (Рис. 4.4)

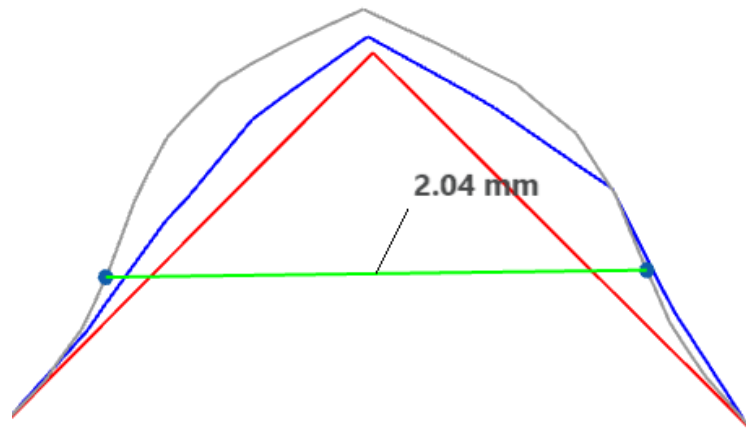


Рисунок 4.4 Результат компенсації інструменту. Розмір сітки 0.25мм, радіус інструменту 1.0мм

Очевидно, що як геометрія, так і радіус кривизни значно покращився. Проте, саме розмір сітки суттєво впливає на швидкодію алгоритму.

Протестуємо ПАК на реальній, цільовій моделі, яка являє собою сканування зубів пацієнта. На Рис 4.5 можна побачити, що в найбільш загострених місцях сканування зубів цільова модель набула округлої форми, відповідно до радіусу інструменту.

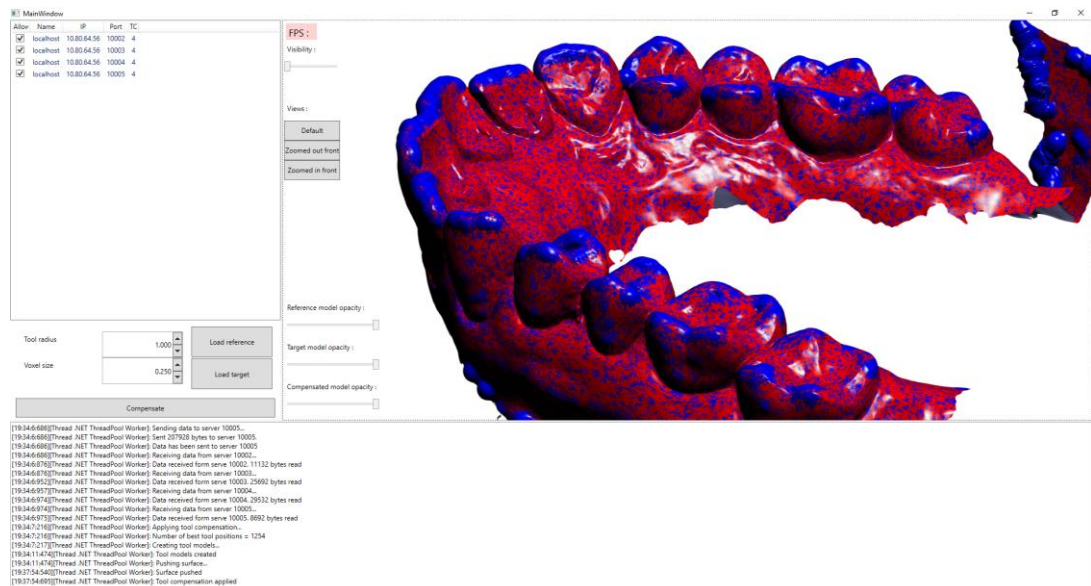


Рисунок 4.5 – Успішне виконання компенсації інструменту на скануванні зубів.

Таку модель можна використовувати як контактну поверхню певного протезу і після виготовлення свердлом з радіусом 1.0мм, він може бути щільно прикладений до зубів пацієнта.

4.2 Визначення ефективності програмно-апаратного комплексу

Визначення ефективності ПАК було виконано шляхом заміру часу виконання операції з різними параметрами та з використанням різної кількості ядер процесора (Таблиця 4.2). Коефіцієнти прискорення $K_p = T_1/T_p$ зведені в таблицю для порівняння (Таблиця 4.1)

Графіки залежності часу виконання від кількості задіяних ядер представлено на Рис. 6-10

Таблиця 4.1 – Порівняння K_p

VoxelSize, ToolRadius	1	2	4	8	16
0.25mm, 1.0mm	1	1.9	3.32	5.15	5.30
0.5mm, 1.0mm	1	1.74	2.73	3.58	4.11
0.25mm, 2.0mm	1	1.9	3.27	5.8	5.72
0.5mm, 2.0mm	1	1.82	2.98	4.88	5.45

Низький приріст K_p при порівнянні 8 ядер і 16 пояснюється типом процесору на якому проводилось тестування. Процесор має лише 8 фізичних ядер. Проте приріст швидкості за рахунок технології HyperThreading достатньо вагомий.

Різниця між максимальним теоретичним K_p і отриманим пояснюється наявністю послідовної частини в алгоритмі, витратами на синхронізацію, та витратами на передачу даних між вузлами.

Таблиця 4.2 – Результати замірів швидкодії ПАК

Method	Cores	VoxelSize	ToolRadius	Mean
BuildToolCompensation	1	0.25	1	11,493.9 ms
BuildToolCompensation	1	0.25	2	39,005.9 ms
BuildToolCompensation	1	0.5	1	877.4 ms
BuildToolCompensation	1	0.5	2	3,051.6 ms
BuildToolCompensation	2	0.25	1	6,034.8 ms
BuildToolCompensation	2	0.25	2	20,498.5 ms
BuildToolCompensation	2	0.5	1	504.9 ms
BuildToolCompensation	2	0.5	2	1,676.2 ms
BuildToolCompensation	4	0.25	1	3,460.8 ms
BuildToolCompensation	4	0.25	2	11,925.2 ms
BuildToolCompensation	4	0.5	1	321.9 ms
BuildToolCompensation	4	0.5	2	1,022.4 ms
BuildToolCompensation	8	0.25	1	2,230.5 ms
BuildToolCompensation	8	0.25	2	7,675.2 ms
BuildToolCompensation	8	0.5	1	245.1 ms
BuildToolCompensation	8	0.5	2	624.9 ms
BuildToolCompensation	16	0.25	1	2,167.9 ms
BuildToolCompensation	16	0.25	2	6,817.9 ms
BuildToolCompensation	16	0.5	1	213.3 ms
BuildToolCompensation	16	0.5	2	560.4 ms

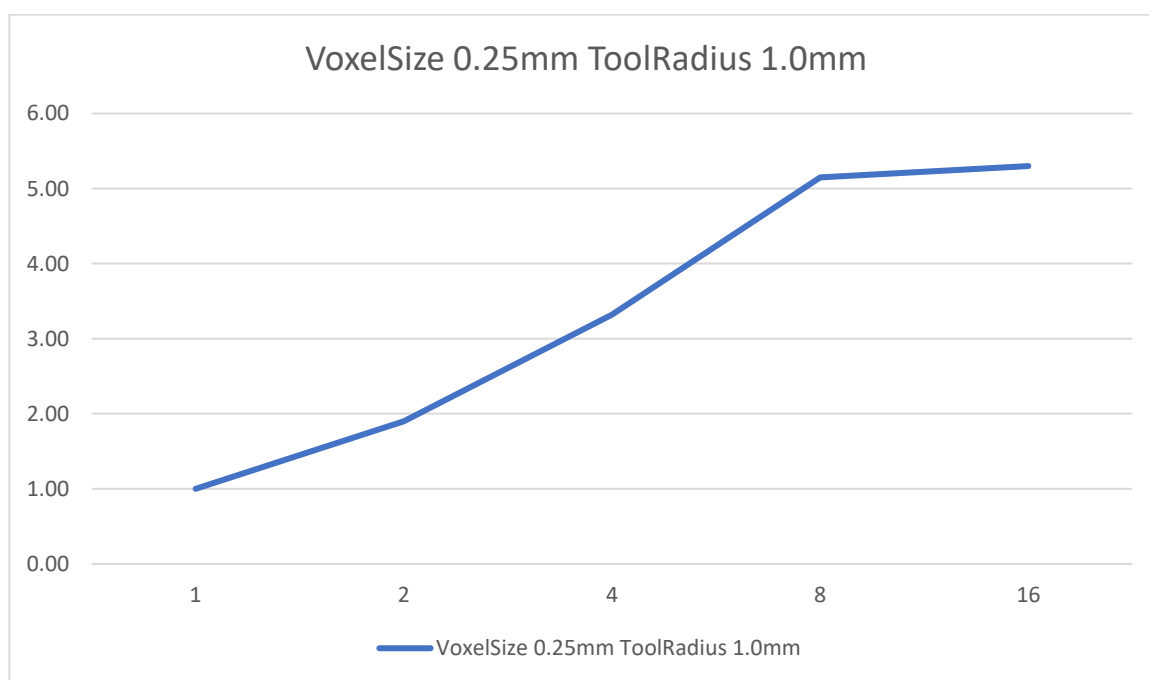


Рисунок 4.6 – Залежність Кп від кількості задіяних ядер. При
VoxelSize = 0.25mm, ToolRadius = 1.0mm

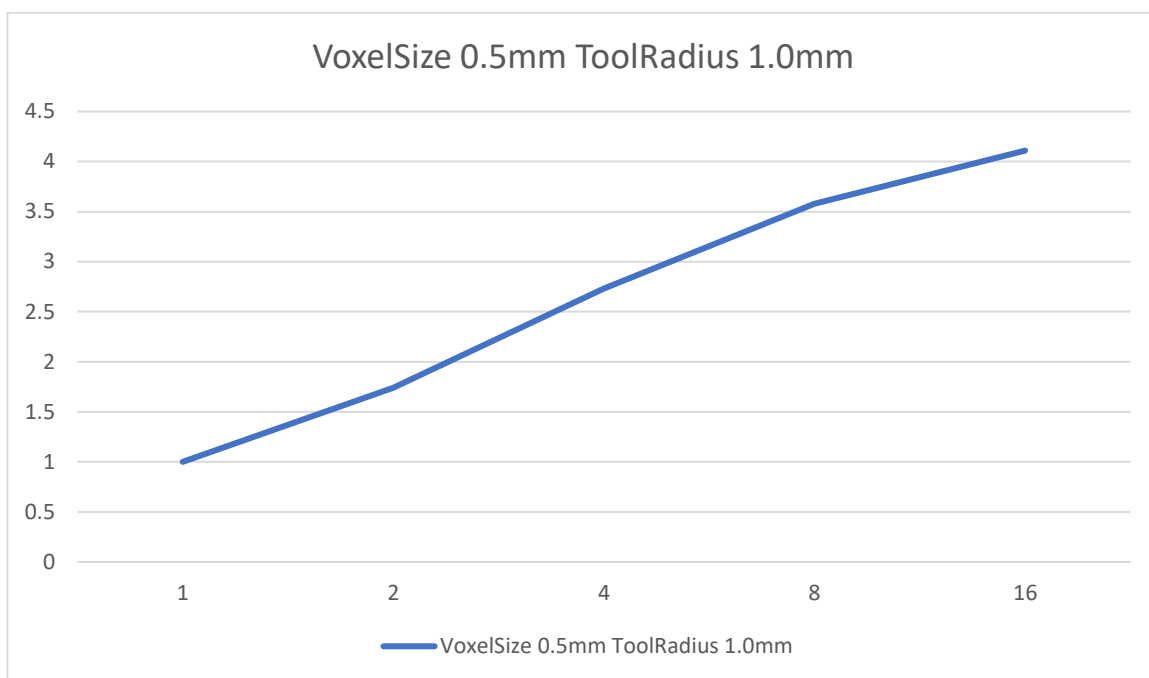


Рисунок 4.7 – Залежність Кп від кількості задіяних ядер. При VoxelSize = 0.5mm, ToolRadius = 1.0mm

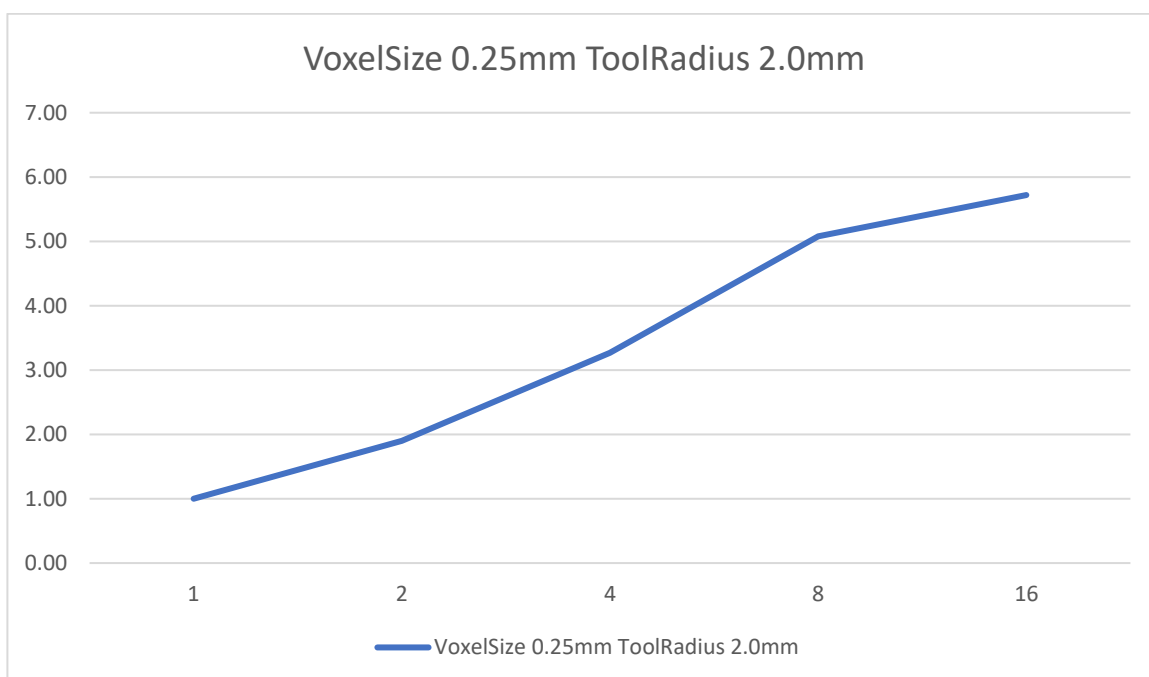


Рисунок 4.8 – Залежність Кп від кількості задіяних ядер. При VoxelSize = 0.25mm, ToolRadius = 2.0mm

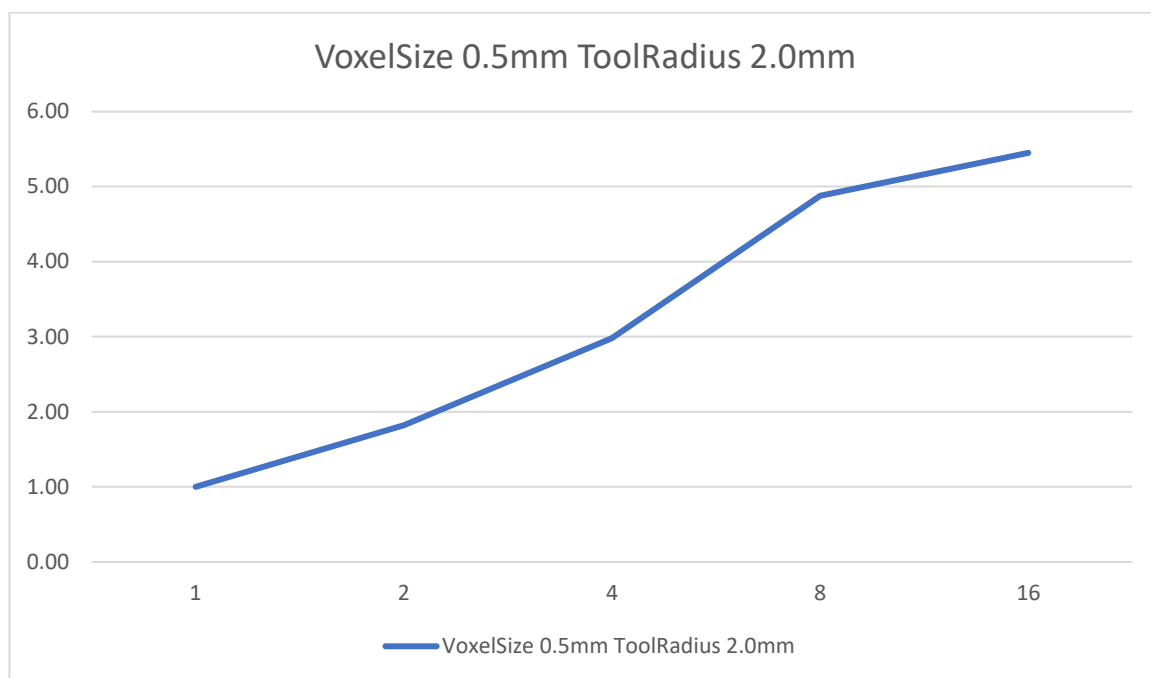


Рисунок 4.9 – Залежність Кп від кількості задіяних ядер. При VoxelSize = 0.5mm, ToolRadius = 2.0mm

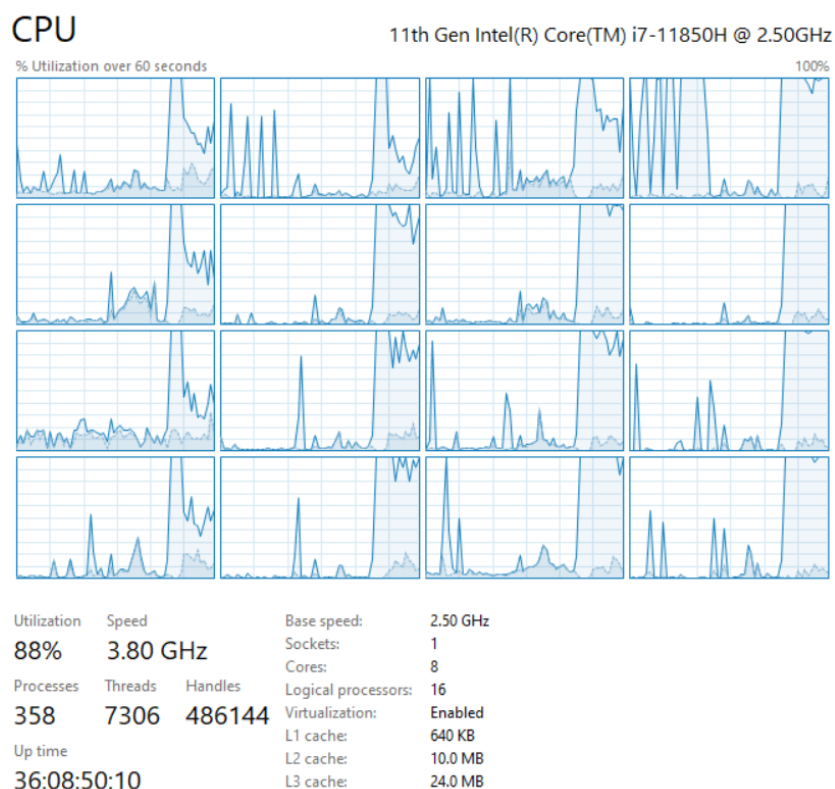


Рисунок 4.10 – Завантаження процесорів під час роботи ПАК

Висновки до розділу 4

В даному розділі було проведене як концептуальне тестування на абстрактній моделі, так і тестування на реальній моделі. Тестування проводилось в режимі симуляції розподіленої системи зі спільною пам'яттю в кожному вузлі. ПАК успішно виконав головну закладену функцію.

Також було проведено тестування швидкодії, побудовані графіки залежності швидкодії від параметрів інструменту так і від точності(розміру сітки). Отриманий коефіцієнт прискорення задовольняє вимогам поставленим для цього проекту.

					ІАЛЦ.466500.003 ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

ОСНОВНІ РЕЗУЛЬТАТИ І ВИСНОВКИ

- В даному проекті було розглянуто задачу компенсації інструменту як приклад операції типової для CAD систем, які використовуються в цифрових стоматологічних лабораторіях. Запропоновано та реалізовано ПАК, метою якого було збільшення точності даної операції при зменшенні часу розрахунку.
- Для апаратної частини ПАК було зроблено огляд технічних рішень та обрано оптимальну конфігурацію для лабораторій середнього розміру. Для лабораторій з кількістю робочих станцій більше 10, рекомендується обрати більш потужну конфігурацію вузла.
- Для реалізації програмної частини ПАК було обрано платформу .NET одної з останніх версій та мову C#. Дана комбінація містить потужні засоби для паралельного програмування. Відмінно підходить як для реалізації клієнтських додатків, так і серверної частини. Потужне ком'юніті та безліч бібліотек дають велику гнучкість при реалізації тих, чи інших компонентів комплексу.
- Тестування ПАК було проведено як на синтетичних моделях, так і на реальних скануваннях зубів пацієнтів. Для наглядної демонстрації та зручності тестування було розроблено прототип CAD системи. Отримані коефіцієнти прискорення задовольняють вимогам. Дано пояснення, чому не вдалося досягти теоретично максимально можливого прискорення.
- В якості розвитку проекту, пропонується розробити планувальник задач для серверної частини, реалізувати можливість розрахунку інших типових операцій.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. What is cutter compensation [Електронний ресурс] – режим доступу:
<https://machmotion.com/blog/what-is-cutter-compensation/>
2. Exocad [Електронний ресурс] – режим доступу:
<https://exocad.com/about-us/about-exocad>
3. 3Shape [Електронний ресурс] – режим доступу:
<https://www.3shape.com/en/our-story>
4. Drill compensation Exocad [Електронний ресурс] – режим доступу:
https://wiki.exocad.com/wiki/index.php/Designing_the_inside_of_the_crown
5. Drill compensation 3Shape [Електронний ресурс] – режим доступу:
<https://www.3shape.com/en/news/2015/10-ways-3shape-made-cadcam-technology-so-much-better>
6. Tooth Preparation Tips for CAD/CAM [Електронний ресурс] – режим доступу:
https://www.youtube.com/watch?v=aInttHr0j_Q
7. Drill compensation for Full Dentrues [Електронний ресурс] – режим доступу:
<https://dentallabnetwork.com/forums/threads/drill-compensation-for-full-dentures.31157/>
8. Parallel computing [Електронний ресурс] – режим доступу:
https://en.wikipedia.org/wiki/Parallel_computing
9. Архітектури КС [Електронний ресурс] – режим доступу:
<https://exocad.com/about-us/about-exocad>
10. Intel [Електронний ресурс] – режим доступу:
<https://www.intel.com/content/www/us/en/homepage.html>
11. Intel Core Processors [Електронний ресурс] – режим доступу:
<https://ark.intel.com/content/www/us/en/ark/products/series/230485/13th-generation-intel-core-i9-processors.html>

12. Intel Xeon Processors [Електронний ресурс] – режим доступу:
<https://ark.intel.com/content/www/us/en/ark/products/series/125191/intel-xeon-scalable-processors.html>
13. AMD [Електронний ресурс] – режим доступу:
<https://www.amd.com/en.html>
14. AMD Ryzen Processors [Електронний ресурс] – режим доступу:
<https://www.amd.com/en/processors/ryzen>
15. AMD EPYC Processors [Електронний ресурс] – режим доступу:
<https://www.amd.com/en/processors/epyc-server-cpu-family>
16. PassMark Software [Електронний ресурс] – режим доступу:
<https://www.passmark.com/>
17. Parallel programming in .NET [Електронний ресурс] – режим доступу:
<https://learn.microsoft.com/en-us/dotnet/standard/parallel-programming/>
18. Task Parallel Library (TPL) [Електронний ресурс] – режим доступу:
<https://learn.microsoft.com/en-us/dotnet/standard/parallel-programming/task-parallel-library-tpl>
19. Introducing to PLINQ [Електронний ресурс] – режим доступу:
<https://learn.microsoft.com/en-us/dotnet/standard/parallel-programming/introduction-to-plinq>
20. Task Scheduler Class [Електронний ресурс] – режим доступу:
<https://learn.microsoft.com/en-us/dotnet/api/system.threading.tasks.taskscheduler?view=net-7.0>
21. ThreadPool Class [Електронний ресурс] – режим доступу:
<https://learn.microsoft.com/en-us/dotnet/api/system.threading.threadpool?view=net-7.0>
22. Overview of synchronization primitives [Електронний ресурс] – режим доступу:
<https://learn.microsoft.com/en-us/dotnet/standard/threading/overview-of-synchronization-primitives>

23.Data Structures for Parallel Programming [Електронний ресурс] – режим доступу:

<https://learn.microsoft.com/en-us/dotnet/standard/parallel-programming/data-structures-for-parallel-programming?source=recommendations>

24.Корочкін О.В. Паралельні та розподілені обчислення: Навч. Посібник, друге видання / І. А. Жуков, О. В. Корочкін, Київ: «Корнійчук», 2014. – 284 с.

25.Socket Class [Електронний ресурс] – режим доступу:

<https://learn.microsoft.com/ru-ru/dotnet/api/system.net.sockets.socket?view=net-8.0>

26.Крутько О.М. Аналіз засобів управління потоками в масштабованих комп'ютерних системах / О.В. Корочкін, О.В. Русанова, О.М. Крутько // Адаптивні системи автоматичного управління : міжвідомчий науково-технічний збірник. - 2022. – № 2 (41). – С. 100-105.

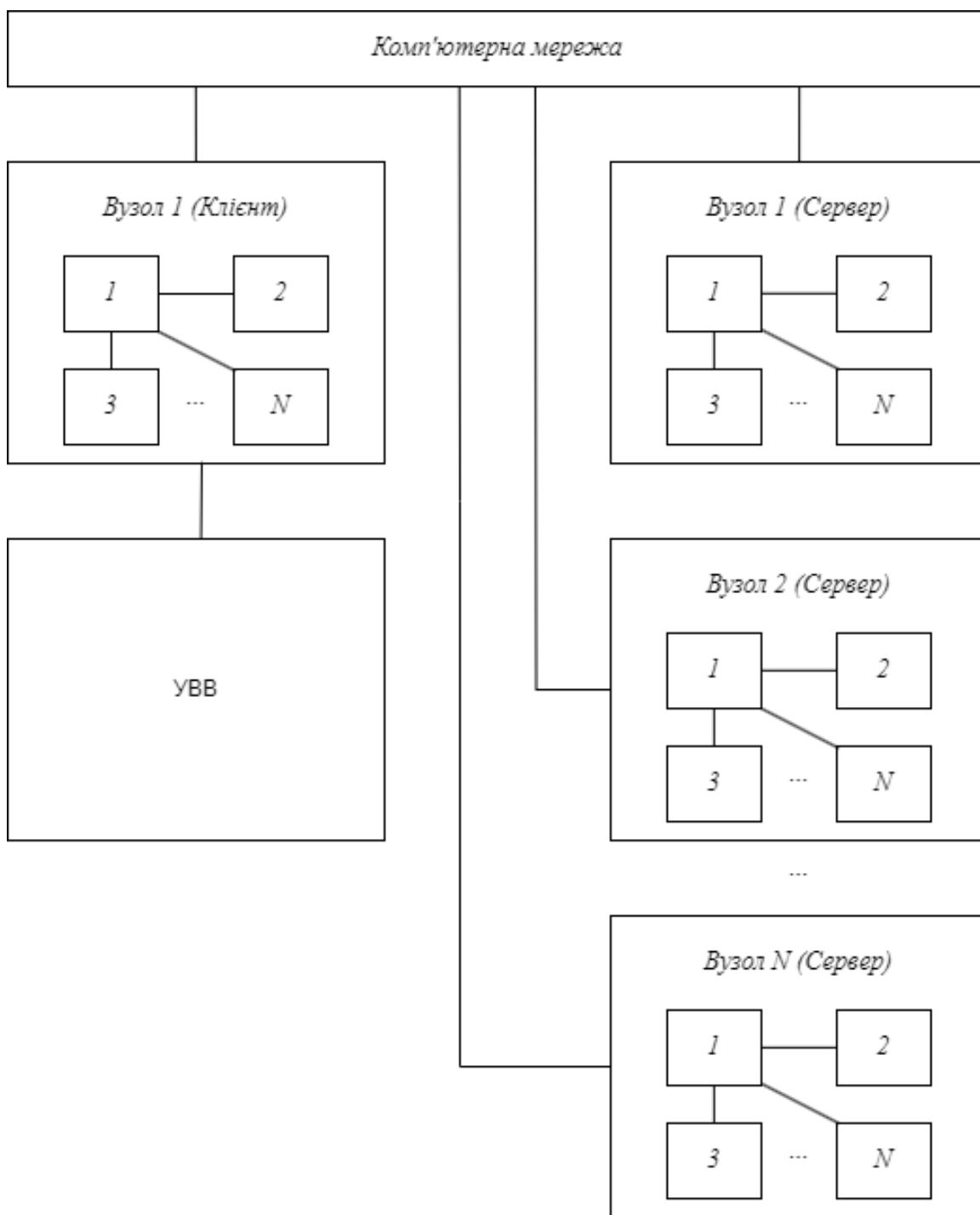
ДОДАТОК А
ДО ДИПЛОМНОГО ПРОЕКТУ

На тему: «Програмно-апаратний комплекс для автоматичної компенсації
інструменту»

Структурна схема ПАК

Аркушів 1

Київ 2023



					ІАЛЦ.466500.004 Д1				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Крутько О.М.			Структурна схема ПАК		Літ.	Арк.	Акрушів
Перевір.		Корочкін О.В.						1	1
Н. Контр.		Виноградов Ю.М.							
Затверд.									
					КПІ ім. Ігоря Сікорського ФІОТ ІО-391				

ДОДАТОК Б

ДО ДИПЛОМНОГО ПРОЕКТУ

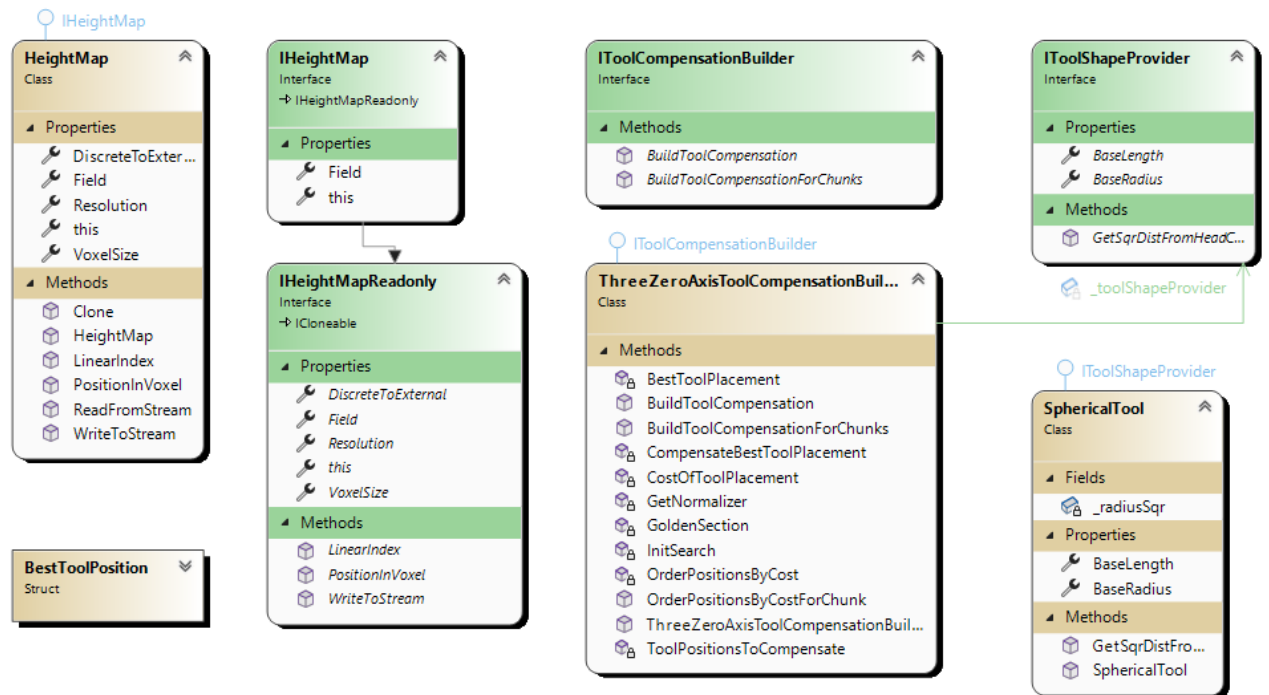
На тему: «Програмно-апаратний комплекс для автоматичної компенсації
інструменту»

Діаграма класів ПАК
(Функціональна схема)

Аркушів 1

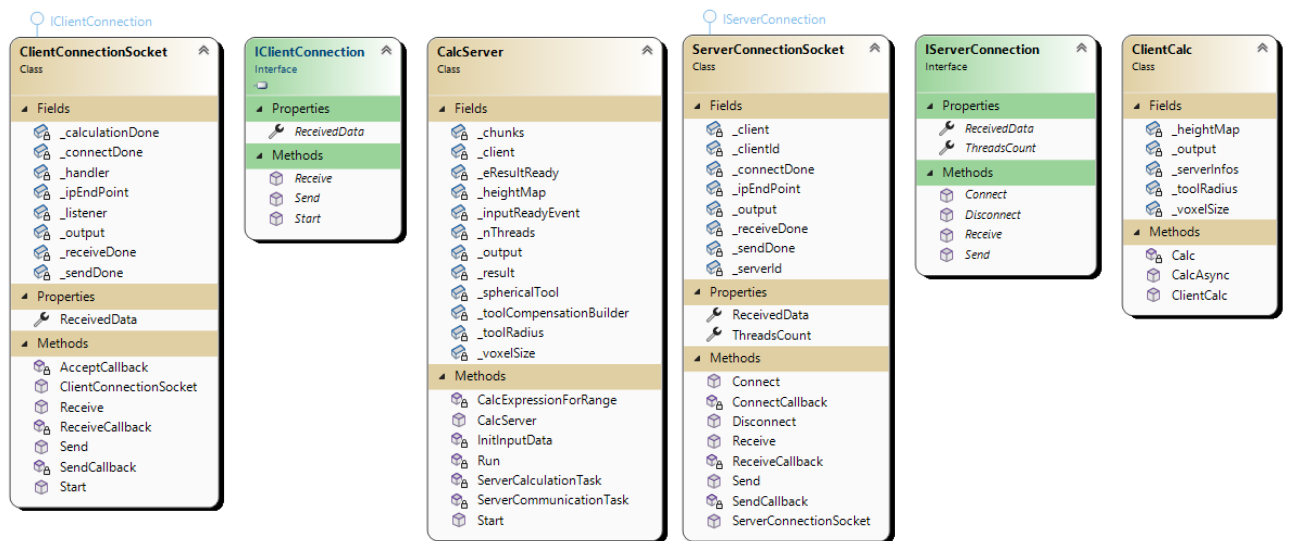
Київ 2023

Компенсация инструмента



Серверна частина

Клієнтська частина



ІАЛЦ.466500.005 Д2

					<i>ІАЛЦ.466500.005 Д2</i>					
Змн.	Арк.	№ докум.	Підпис	Дата						
Розроб.		Крутько О.М.			<i>Діаграма класів ПАК</i> <i>(Функціональна схема)</i>			Лім.	Арк.	Акрушів
Перевір.		Корочкін О.В.							1	1
Н. Контр.		Виноградов Ю.М.						<i>КПІ ім. Ігоря Сікорського</i> <i>ФІОТ ІО-391</i>		
Затверд.										

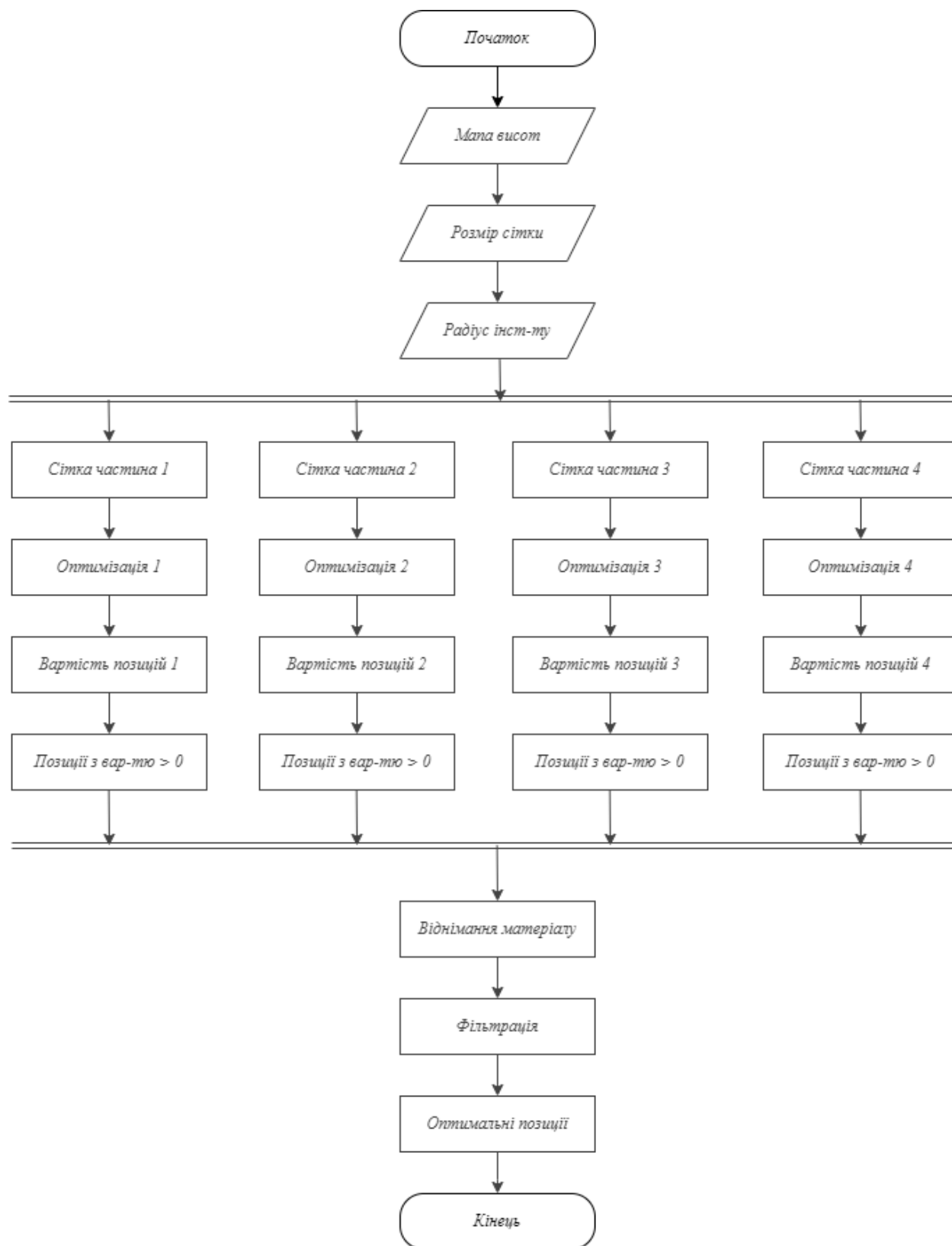
ДОДАТОК В
ДО ДИПЛОМНОГО ПРОЕКТУ

На тему: «Програмно-апаратний комплекс для автоматичної компенсації
інструменту»

Блок-схема алгоритму роботи потоків

Аркушів 1

Київ 2023



					ІАЛЦ.466500.006 ДЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Крутько О.М.			Блок-схема алгоритму роботи потоків		
Перевір.		Корочкін О.В.					
Н. Контр.		Виноградов Ю.М.					
Затверд.							
						Літ.	Арк.
							Акрушів
						1	1
						КПІ ім. Ігоря Сікорського	
						ФІОТ ІО-391	

ДОДАТОК Г
ДО ДИПЛОМНОГО ПРОЕКТУ

На тему: «Програмно-апаратний комплекс для автоматичної компенсації
інструменту»

Процес взаємодії Клієнта і Серверів

(Принципова схема)

Аркушів 1

Київ 2023

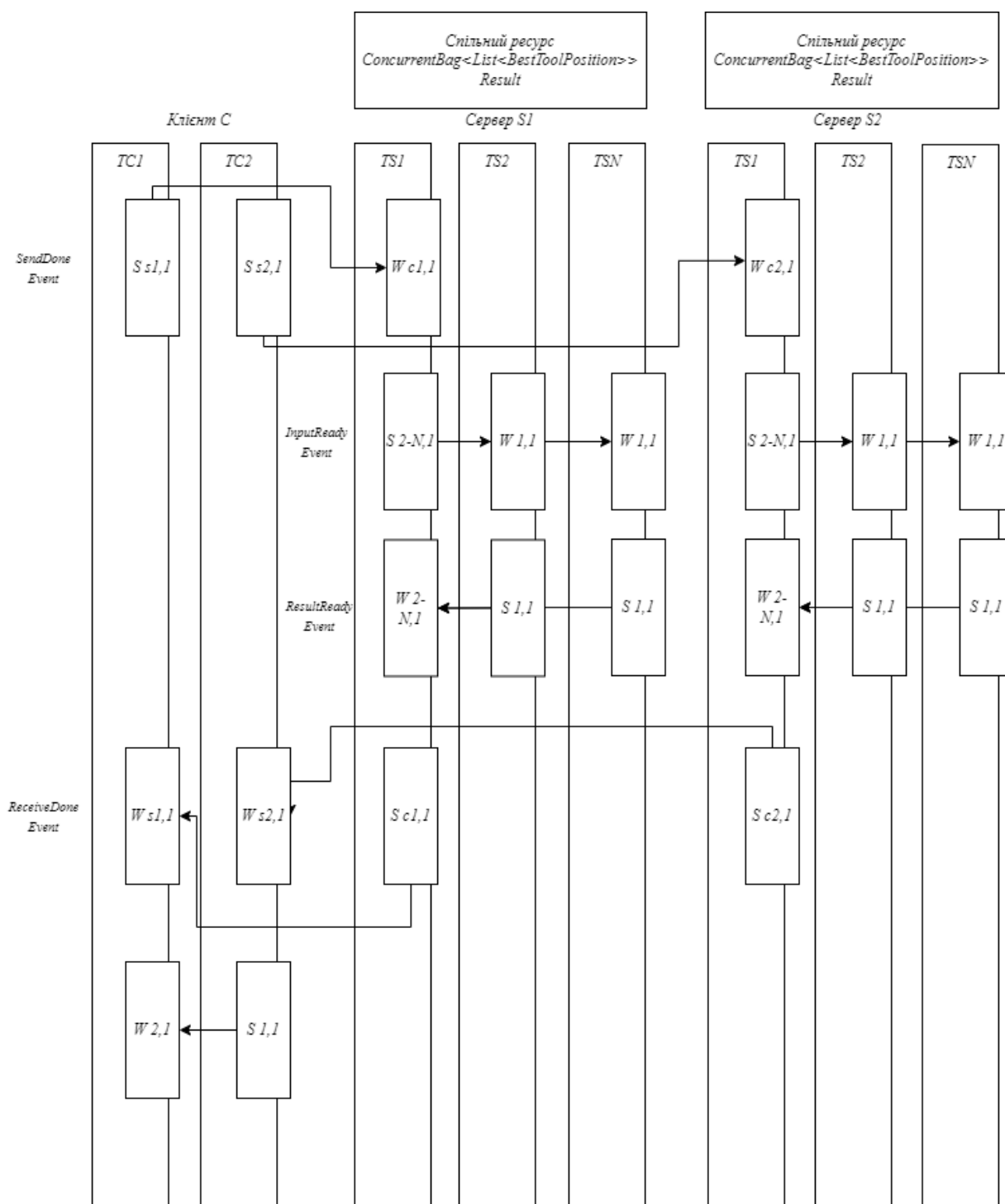
ДОДАТОК Д
ДО ДИПЛОМНОГО ПРОЕКТУ

На тему: «Програмно-апаратний комплекс для автоматичної компенсації
інструменту»

Процес взаємодії потоків в вузлах

Аркушів 1

Київ 2023



ІАЛЦ.466500.008 Д5

Змн.	Арк.	№ докум.	Підпис	Дата
Розроб.		Крутько О.М.		
Перевір.		Корочкін О.В.		
Н. Контр.		Виноградов Ю.М.		
Затверд.				

Процес взаємодії потоків в вузлах

Лім.	Арк.	Акрушів
	1	1
КПІ ім. Ігоря Сікорського ФІОТ ІО-391		

ДОДАТОК Е

ДО ДИПЛОМНОГО ПРОЕКТУ

На тему: «Програмно-апаратний комплекс для автоматичної компенсації
інструменту»

Текст програми

Аркушів 28

Київ 2023

```

1. // HeightMap.cs
2. // Tool compensation. Distributed system.
3. // Bachelor thesis
4. // Oleksii Krutko. IO-z91
5. // 29.05.2023
6.
7. using System.Numerics;
8. using System.Runtime.CompilerServices;
9.
10. namespace ToolCompensation
11. {
12.     /// <summary>
13.     /// Helper class for book-keeping distance fields
14.     /// </summary>
15.     public class HeightMap : IHeightMap
16.     {
17.         /// <summary>
18.         /// The field
19.         /// </summary>
20.         public List<float> Field { get; }
21.
22.         /// <summary>
23.         /// The resolution
24.         /// </summary>
25.         public Vector2i Resolution { get; }
26.
27.         /// <summary>
28.         /// The voxel size
29.         /// </summary>
30.         public float VoxelSize { get; }
31.
32.         /// <summary>
33.         /// Gets the discrete to external transformation
34.         /// </summary>
35.         public Matrix4x4 DiscreteToExternal { get; }
36.
37.         /// <summary>
38.         /// Constructor
39.         /// </summary>
40.         public HeightMap(Vector2i resolution, float voxelSize, Matrix4x4
discreteToExternal)
41.         {
42.             Field = new float[resolution.X * resolution.Y].ToList();
43.             Resolution = resolution;
44.             VoxelSize = voxelSize;
45.             DiscreteToExternal = discreteToExternal;
46.         }
47.
48.         /// <summary>
49.         /// Indexer access to the data.
50.         /// </summary>
51.         public float this[int x, int y]
52.         {
53.             get => Field[LinearIndex(x, y)];
54.             set => Field[LinearIndex(x, y)] = value;
55.         }
56.
57.         /// <summary>
58.         /// Helping method to convert x,y to a position in the 1D array.
59.         /// </summary>
60.         /// <returns></returns>
61.         [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public int LinearIndex(int x, int y)

```

					ІАЛЦ.466500.009 Д6		
Змн.	Арк.	№ докум.	Підпис	Дата	Текст програми		
Розроб.		Крутько О.М.					
Перевір.		Корочкін О.В.					
Н. Контр.		Виноградов Ю.М.					
Затверд.							
					Лім.	Арк.	Акрушів
						1	28
					КПІ ім. Ігоря Сікорського ФІОТ ІО-391		

```

62.
63.     {
64.         return x + y * Resolution.X;
65.     }
66.
67.     /// <summary>
68.     /// Converts linear index to a position in the voxel field in place.
69.     /// No Checks if the voxel field contains the point
70.     /// </summary>
71.     /// <param name="idx"></param>
72.     /// <param name="x"></param>
73.     /// <param name="y"></param>
74.     /// <returns></returns>
75.     public void PositionInVoxel(int idx, out int x, out int y)
76.     {
77.         var noZ = idx % (Resolution.X * Resolution.Y);
78.         y = noZ / Resolution.X;
79.         x = noZ % Resolution.X;
80.     }
81.
82.     /// <summary> Clones the height map </summary>
83.     /// <returns>Cloned object</returns>
84.     public object Clone()
85.     {
86.         var clone = new HeightMap(this.Resolution, this.VoxelSize,
this.DiscreteToExternal);
87.         for (var x = 0; x < clone.Resolution.X; x++)
88.         {
89.             for (var y = 0; y < clone.Resolution.Y; y++)
90.             {
91.                 clone[x, y] = this[x, y];
92.             }
93.         }
94.
95.         return clone;
96.     }
97.
98.     /// <summary>
99.     /// Reads height map from stream
100.    /// </summary>
101.    /// <param name="reader"> Reader connected with stream</param>
102.    /// <returns>Interface to height map</returns>
103.    public static IHeightMapReadOnly ReadFromStream(BinaryReader reader)
104.    {
105.        var resX = reader.ReadInt32();
106.        var resY = reader.ReadInt32();
107.
108.        var voxelSize = reader.ReadSingle();
109.
110.        var matrix = new Matrix4x4
111.        {
112.            M11 = reader.ReadSingle(),
113.            M12 = reader.ReadSingle(),
114.            M13 = reader.ReadSingle(),
115.            M14 = reader.ReadSingle(),
116.            M21 = reader.ReadSingle(),
117.            M22 = reader.ReadSingle(),
118.            M23 = reader.ReadSingle(),
119.            M24 = reader.ReadSingle(),
120.            M31 = reader.ReadSingle(),
121.            M32 = reader.ReadSingle(),
122.            M33 = reader.ReadSingle(),
123.            M34 = reader.ReadSingle(),
124.            M41 = reader.ReadSingle(),
125.            M42 = reader.ReadSingle(),
126.            M43 = reader.ReadSingle(),
127.            M44 = reader.ReadSingle()

```

					ІАЛЦ.466500.009 Д6	Арк.
						2
Змн.	Арк.	№ докум.	Підпис	Дата		

```

128.         };
129.
130.         var result = new HeightMap(new Vector2i(resX, resY), voxelSize, matrix);
131.
132.         for (var i = 0; i < result.Field.Count; i++)
133.             result.Field[i] = reader.ReadSingle();
134.
135.         return result;
136.     }
137.
138.     /// <inheritdoc />
139.     public void WriteToStream(BinaryWriter writer)
140.     {
141.         //Resolution
142.         writer.Write(Resolution.X);
143.         writer.Write(Resolution.Y);
144.
145.         // Voxel size
146.         writer.Write(VoxelSize);
147.
148.         // DiscreteToExternal matrix
149.         writer.Write(DiscreteToExternal.M11);
150.         writer.Write(DiscreteToExternal.M12);
151.         writer.Write(DiscreteToExternal.M13);
152.         writer.Write(DiscreteToExternal.M14);
153.         writer.Write(DiscreteToExternal.M21);
154.         writer.Write(DiscreteToExternal.M22);
155.         writer.Write(DiscreteToExternal.M23);
156.         writer.Write(DiscreteToExternal.M24);
157.         writer.Write(DiscreteToExternal.M31);
158.         writer.Write(DiscreteToExternal.M32);
159.         writer.Write(DiscreteToExternal.M33);
160.         writer.Write(DiscreteToExternal.M34);
161.         writer.Write(DiscreteToExternal.M41);
162.         writer.Write(DiscreteToExternal.M42);
163.         writer.Write(DiscreteToExternal.M43);
164.         writer.Write(DiscreteToExternal.M44);
165.
166.         var byteArray = new byte[Field.Count * sizeof(float)];
167.         Buffer.BlockCopy(Field.ToArray(), 0, byteArray, 0, byteArray.Length);
168.         writer.Write(byteArray);
169.     }
170. }
171. }

```

```

1.  // IHeightMap.cs
2.  // Tool compensation. Distributed system.
3.  // Bachelor thesis
4.  // Oleksii Krutko. IO-z91
5.  // 29.05.2023
6.
7.  namespace ToolCompensation
8.  {
9.      /// <summary>
10.     /// Writable interface for HeightMap
11.     /// </summary>
12.     public interface IHeightMap: IHeightMapReadOnly
13.     {
14.         /// <summary>The field</summary>
15.         new List<float> Field { get; }
16.
17.         /// <summary>
18.         /// Indexer access to the data.
19.         /// </summary>
20.         new float this[int x, int y] { get; set; }
21.     }

```

					ІАЛЦ.466500.009 Д6	Арк.
						3
Змн.	Арк.	№ докум.	Підпис	Дата		

```

22. }

1. // IHeightMapReadOnly.cs
2. // Tool compensation. Distributed system.
3. // Bachelor thesis
4. // Oleksii Krutko. IO-z91
5. // 29.05.2023
6.
7. using System.Numerics;
8.
9. namespace ToolCompensation
10. {
11.     /// <summary>ReadOnly interface for HeightMap</summary>
12.     public interface IHeightMapReadOnly: ICloneable
13.     {
14.         /// <summary>Gets the discrete to external.</summary>
15.         Matrix4x4 DiscreteToExternal { get; }
16.
17.         /// <summary>The voxel size</summary>
18.         float VoxelSize { get; }
19.
20.         /// <summary>The field</summary>
21.         List<float> Field { get; }
22.
23.         /// <summary>The resolution</summary>
24.         Vector2i Resolution { get; }
25.
26.         /// <summary>Indexer access to the data.</summary>
27.         float this[int x, int y] { get; }
28.
29.         /// <summary> Helping method to convert x,y to a position in the 1D array.
30.         </summary>
31.         /// <returns></returns>
32.         int LinearIndex(int x, int y);
33.
34.         /// <summary>
35.         /// Converts linear index to a position in the voxel field in place.
36.         /// No Checks if the voxel field contains the point
37.         /// </summary>
38.         /// <param name="idx"></param>
39.         /// <param name="x"></param>
40.         /// <param name="y"></param>
41.         /// <returns></returns>
42.         void PositionInVoxel(int idx, out int x, out int y);
43.
44.         /// <summary>Writes height map to stream.</summary>
45.         void WriteToStream(BinaryWriter writer);
46.     }
}

1. // BestToolPosition.cs
2. // Tool compensation. Distributed system.
3. // Bachelor thesis
4. // Oleksii Krutko. IO-z91
5. // 29.05.2023
6.
7. using System.Numerics;
8.
9. namespace ToolCompensation
10. {
11.     /// <summary>
12.     /// Structure represents a single tool position that is supposed to be subtracted
13.     from
14.     /// the target surface while applying tool compensation.
15.     /// </summary>
16.     public struct BestToolPosition
17.     {

```

					ІАЛЦ.466500.009 Д6	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

```

17.      /// <summary>
18.      ///   The best cost
19.      /// </summary>
20.      public float BestCost;
21.      /// <summary>
22.      ///   The identifier
23.      /// </summary>
24.      public int Id;
25.      /// <summary>
26.      ///   The best position
27.      /// </summary>
28.      public Vector3 BestPos;
29.
30.      /// <summary>
31.      /// constructor
32.      /// </summary>
33.      public BestToolPosition(float bestCost, int id, Vector3 bestPos)
34.      {
35.          BestCost = bestCost;
36.          Id = id;
37.          BestPos = bestPos;
38.      }
39.  }
40. }

1.  // IToolCompensationBuilder.cs
2.  // Tool compensation. Distributed system.
3.  // Bachelor thesis
4.  // Oleksii Krutko. IO-z91
5.  // 29.05.2023
6.
7.  namespace ToolCompensation;
8.
9.  /// <summary>
10. /// Interface for tool positions generators. Such as: 3+0-axis, 3+1-axis, 3+2-axis, 5-
    axis etc.
11. /// </summary>
12. public interface IToolCompensationBuilder
13. {
14.     /// <summary>
15.     /// Builds tool compensation
16.     /// </summary>
17.     IEnumerable<BestToolPosition> BuildToolCompensation(IHeightMapReadOnly heightMap,
        bool parallel);
18.
19.     /// <summary>
20.     /// Builds tool compensation for chunks
21.     /// </summary>
22.     IEnumerable<BestToolPosition> BuildToolCompensationForChunks(IHeightMapReadOnly
        heightMap,
23.         IEnumerable<IEnumerable<BestToolPosition>> chunks);
24. }

1.  // IToolShapeProvider.cs
2.  // Tool compensation. Distributed system.
3.  // Bachelor thesis
4.  // Oleksii Krutko. IO-z91
5.  // 01.03.2023
6.
7.  namespace ToolCompensation
8.  {
9.      /// <summary>
10.     /// Interface for tool shapes. Such as: simple spherical, conic etc.
11.     /// </summary>
12.     public interface IToolShapeProvider
13.     {
14.         /// <summary> Gets tool base radius. </summary>

```

					ІАЛЦ.466500.009 Д6	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		5

```

15.         float BaseRadius { get; }
16.
17.         /// <summary> Gets tool base length. </summary>
18.         float BaseLength { get; }
19.
20.         /// <summary>
21.         /// Gets distance from head center to its surface in specific direction
22.         /// </summary>
23.         /// <param name="angle">Angle of direction. [0:PI]</param>
24.         /// <returns></returns>
25.         float GetSqrDistFromHeadCenterToHeadSurface(float angle);
26.     }
27. }

1. // ThreeZeroAxisToolCompensationBuilder.cs
2. // Tool compensation. Distributed system.
3. // Bachelor thesis
4. // Oleksii Krutko. IO-z91
5. // 29.05.2023
6.
7. using System.Numerics;
8.
9. namespace ToolCompensation;
10.
11. /// <summary>
12. /// Implementation 3+0-axis tool compensation builder.
13. /// </summary>
14. public class ThreeZeroAxisToolCompensationBuilder: IToolCompensationBuilder
15. {
16.     private readonly IToolShapeProvider _toolShapeProvider;
17.
18.     /// <summary>
19.     /// constructor
20.     /// <param name="toolShapeProvider">Tool shape</param>
21.     /// </summary>
22.     public ThreeZeroAxisToolCompensationBuilder(
23.         IToolShapeProvider toolShapeProvider)
24.     {
25.         _toolShapeProvider = toolShapeProvider;
26.     }
27.
28.     /// <summary>
29.     /// Optimizes objective function along a parametrized line in direction of a
30.     /// </summary>
31.     /// <param name="of">Objective function</param>
32.     /// <param name="dv">Gradient</param>
33.     /// <param name="max">Longest allowed step</param>
34.     /// <param name="fVal"></param>
35.     /// <param name="precision">Stopping criteria as proportion of longest step</param>
36.     /// <returns>True if we improved</returns>
37.     private static bool GoldenSection(
38.         Func<float, float, float> of,
39.         ref Vector2 dv,
40.         float max,
41.         out float fVal,
42.         float precision = 1e-4f)
43.     {
44.         const float gd = 1.618034f;
45.         var accuracy = precision * max;
46.         var improved = false;
47.
48.         // Notation follows a subdivision of regions as [a c d b], where of(c) and of(d)
49.         // then the region is updated to either [a n c d], or [c d n b], and parameters
50.         // are renamed to

```

					ІАЛЦ.466500.009 Д6	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

```

50.      // [a c d b] for next iteration. See any reference on golden section search for
      further explanation.
51.      var a = 0.0f;
52.      var b = max;
53.      var c = b - (b - a) / gd;
54.      var d = a + (b - a) / gd;
55.
56.      // fa
57.      var fa = of(0.0f, 0.0f);
58.      fVal = fa;
59.
60.      // fc
61.      var vcX = dv.X * c;
62.      var vcY = dv.Y * c;
63.      var fc = of(vcX, vcY);
64.
65.      // fd
66.      var vdX = dv.X * d;
67.      var vdY = dv.Y * d;
68.      var fd = of(vdX, vdY);
69.
70.      // fb, we omit this as
71.      while (Math.Abs(b - a) > accuracy && Math.Abs(fc - fd) > 1e-6f)
72.      {
73.          if (fc < fd)
74.          {
75.              b = d;
76.              d = c;
77.              fd = fc;
78.              c = b - (b - a) / gd;
79.              vcX = dv.X * c;
80.              vcY = dv.Y * c;
81.              fc = of(vcX, vcY);
82.              improved = fd < fVal;
83.          }
84.          else
85.          {
86.              a = c;
87.              c = d;
88.              fc = fd;
89.              d = a + (b - a) / gd;
90.              vdX = dv.X * d;
91.              vdY = dv.Y * d;
92.              fd = of(vdX, vdY);
93.              improved = fc < fVal;
94.          }
95.      }
96.
97.      if (!improved)
98.          return false;
99.
100.     var step = (a + b) / 2;
101.     dv.X *= step;
102.     dv.Y *= step;
103.     fVal = of(dv.X, dv.Y);
104.
105.     return true;
106. }
107.
108. /// <summary>
109. /// Calculates the cost of a specific tool placement and
110. /// finds the base Height for this position
111. /// </summary>
112. /// <param name="targetHeight"></param>
113. /// <param name="compMap">Current tool depth at each voxel location</param>
114. /// <param name="centre">coordinates in voxel space of current voxel to
    process</param>

```

					ІАЛЦ.466500.009 Д6	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

```

115.    /// <param name="offsetX">offset in real coordinates from centre</param>
116.    /// <param name="offsetY">offset in real coordinates from centre</param>
117.    /// <param name="toolMap">Function that takes squared distance from centre and
    returns height</param>
118.    /// <param name="toolRadius">radius of tool</param>
119.    /// <param name="height">How high the toolMap = 0 level is (for flathead it is tip,
    for sphere head it is
120.    /// base of sphere)</param>
121.    /// <returns>cost as accumulated violation in voxel grid</returns>
122.    private static float CostOfToolPlacement(
123.        float targetHeight,
124.        IHeightMapReadOnly compMap,
125.        Vector2i centre,
126.        float offsetX,
127.        float offsetY,
128.        Func<float, float> toolMap,
129.        float toolRadius,
130.        out float height)
131.    {
132.        height = targetHeight;
133.
134.        if (offsetX * offsetX + offsetY * offsetY > toolRadius * toolRadius)
135.            return float.PositiveInfinity;
136.
137.        height -= toolMap((offsetX * offsetX) + (offsetY * offsetY));
138.
139.        var cost = GetNormalizer(compMap, centre, offsetX, offsetY, toolMap,
    toolRadius, height, out var normalizer);
140.
141.        if (normalizer == 0)
142.            return float.PositiveInfinity;
143.
144.        return cost / normalizer;
145.    }
146.
147.    /// <summary>
148.    /// Determines how the placement aligns to grid and the distance to tool centre is
    in each point
149.    /// </summary>
150.    private static float GetNormalizer(IHeightMapReadOnly compMap, Vector2i centre,
151.        float offsetX,
152.        float offsetY,
153.        Func<float, float> toolMap,
154.        float toolRadius, float height, out int normalizer)
155.    {
156.        var posX = centre.X * compMap.VoxelSize + offsetX;
157.        var posY = centre.Y * compMap.VoxelSize + offsetY;
158.
159.        var minX = (int)((posX - toolRadius) / compMap.VoxelSize);
160.        var maxX = (int)((posX + toolRadius) / compMap.VoxelSize) + 1;
161.        var minY = (int)((posY - toolRadius) / compMap.VoxelSize);
162.        var maxY = (int)((posY + toolRadius) / compMap.VoxelSize) + 1;
163.
164.        // truncate
165.        minX = minX < 0 ? 0 : minX;
166.        maxX = maxX > compMap.Resolution.X ? compMap.Resolution.X : maxX;
167.        minY = minY < 0 ? 0 : minY;
168.        maxY = maxY > compMap.Resolution.Y ? compMap.Resolution.Y : maxY;
169.
170.        var radius2 = toolRadius * toolRadius;
171.
172.        var cost = 0.0f;
173.        normalizer = 0;
174.
175.        for (var y = minY; y < maxY; ++y)
176.        {
177.            var yf = (y * compMap.VoxelSize - posY);

```

					ІАЛЦ.466500.009 Д6	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

```

178.
179.         for (int x = minX, linIdx = compMap.LinearIndex(minX, y); x < maxX; ++x,
++linIdx)
180.         {
181.             var xf = (x * compMap.VoxelSize - posX);
182.             var dist = xf * xf + yf * yf;
183.
184.             if (dist > radius2)
185.                 continue;
186.
187.             var depth = toolMap(dist) + height;
188.             var diff = depth - compMap.Field[linIdx];
189.
190.             if (float.IsNaN(diff))
191.                 continue;
192.
193.             cost += diff > 0 ? diff : 0;
194.             normalizer += 1;
195.         }
196.     }
197.
198.     return cost;
199. }
200.
201.     /// <summary>
202.     /// Calculates best tool placements using numerical derivative, gradient descent
with momentum and
203.     /// golden section search
204.     /// </summary>
205.     /// <param name="targetId"></param>
206.     /// <param name="compMap">Current tooled heights</param>
207.     /// <param name="centre">voxel to tool</param>
208.     /// <param name="toolMap">shape of tool</param>
209.     /// <param name="toolRadius">radius of tool</param>
210.     /// <param name="bestCost">cost</param>
211.     /// <returns>Optimal position</returns>
212.     private static Vector3 BestToolPlacement(int targetId, IHeightMapReadOnly compMap,
Vector2i centre,
213.         Func<float, float> toolMap, float toolRadius, out float bestCost)
214.     {
215.         const int maxOptimizationSteps = 50;
216.         const float momentum = 0.05f;
217.
218.         var searchHeight = InitSearch(toolRadius, out var searchHeightX, out var
searchHeightY, out var bestOffsetX, out var bestOffsetY);
219.
220.         bestCost = CostOfToolPlacement(compMap.Field[targetId], compMap, centre,
bestOffsetX, bestOffsetY, toolMap, toolRadius, out var bestHeight);
221.
222.         // we use some momentum in optimization
223.         if (Math.Abs(bestCost) <= float.Epsilon)
224.         {
225.             return new Vector3(centre.X * compMap.VoxelSize, centre.Y *
compMap.VoxelSize, bestHeight);
226.         }
227.
228.         for (var iOffY = -5; iOffY <= 5; ++iOffY)
229.         {
230.             for (var iOffX = -5; iOffX <= 5; ++iOffX)
231.             {
232.                 var thisOffsetX = iOffX * toolRadius / 6;
233.                 var thisOffsetY = iOffY * toolRadius / 6;
234.
235.                 var cost = CostOfToolPlacement(compMap.Field[targetId], compMap,
centre,
236.                     iOffX * toolRadius / 6, iOffY * toolRadius / 6, toolMap,
toolRadius, out var thisHeight);

```

```

237.         if (cost > bestCost)
238.             continue;
239.
240.         bestCost = cost;
241.         bestOffsetX = thisOffsetX;
242.         bestOffsetY = thisOffsetY;
243.         bestHeight = thisHeight;
244.
245.         if (!(bestCost < 1e-6f))
246.             continue;
247.
248.         bestCost = 0;
249.
250.         return new Vector3(bestOffsetX + centre.X * compMap.VoxelSize,
bestOffsetY + centre.Y * compMap.VoxelSize,
251.             bestHeight);
252.     }
253. }
254.
255. var gradient = Vector2.Zero;
256.
257. float height;
258.
259. // Construct the objective function in current offset
260. float Of(float vX, float vY)
261. {
262.     return CostOfToolPlacement(
263.         compMap.Field[targetId],
264.         compMap,
265.         centre,
266.         bestOffsetX + vX,
267.         bestOffsetY + vY,
268.         toolMap,
269.         toolRadius,
270.         out height);
271. }
272.
273. // we restrict upper search range quite severely
274. for (var i = 0; i < maxOptimizationSteps; ++i)
275. {
276.     // calculate gradient
277.     var dfX = (Of(searchHeightX.X, searchHeightX.Y) - Of(-searchHeightX.X, -
searchHeightX.Y)) / (2 * searchHeight);
278.     var dfY = (Of(searchHeightY.X, searchHeightY.Y) - Of(-searchHeightY.X, -
searchHeightY.Y)) / (2 * searchHeight);
279.
280.     if (float.IsInfinity(dfX))
281.         dfX = 0;
282.
283.     if (float.IsInfinity(dfY))
284.         dfY = 0;
285.
286.     gradient.X = (1.0f - momentum) * dfX + momentum * gradient.X;
287.     gradient.Y = (1.0f - momentum) * dfY + momentum * gradient.Y;
288.
289.     // we now have a line optimization over t in off + t* grad
290.     if (gradient.LengthSquared() < 1e-12f)
291.     {
292.         // reached stationary point
293.         break;
294.     }
295.
296.     var negGradient = - Vector2.Normalize(gradient);
297.     var maxT = toolRadius / 20;
298.
299.     if (!GoldenSection(Of, ref negGradient, maxT, out var cost))
300.         break;

```

```

301.
302.         bestOffsetX += negGradient.X;
303.         bestOffsetY += negGradient.Y;
304.         bestHeight = height;
305.         bestCost = cost;
306.
307.         if (!(bestCost < 1e-8f))
308.             continue;
309.
310.         // to small to matter
311.         bestCost = 0;
312.         break;
313.     }
314.
315.     return new Vector3(bestOffsetX + centre.X * compMap.VoxelSize,
316.                        bestOffsetY + centre.Y * compMap.VoxelSize,
317.                        bestHeight);
318. }
319.
320. /// <summary>
321. /// Initialize search
322. /// typical tool is .5 mm and tool moves in 20 microns, i.e. 0.02
323. /// </summary>
324. private static float InitSearch(float toolRadius,
325.                                out Vector2 hX, out Vector2 hY, out float bestOffsetX, out float bestOffsetY)
326. {
327.     var searchHeight = toolRadius / 50;
328.     hX = new Vector2(searchHeight, 0);
329.     hY = new Vector2(0, searchHeight);
330.
331.     bestOffsetX = 0.0f;
332.     bestOffsetY = 0.0f;
333.     return searchHeight;
334. }
335.
336. /// <summary>
337. /// Calculates best local tool position, as well as how deep it has to penetrate,
as a vector
338. /// with the z,y, and z position of the tool below head
339. /// </summary>
340. /// <param name="targetId"></param>
341. /// <param name="compMap">the tool compensation map, adjusted during
optimization</param>
342. /// <param name="centre">input centre coordinate, i.e. violation point</param>
343. /// <param name="toolMap">Should take sqr dist from tool centre and return height
of head</param>
344. /// <param name="toolRadius"></param>
345. /// <param name="bestCost">best cost found during optimization</param>
346. /// <returns>Tool position x,y,z, x,y, is center of tool, z position of base of
head, unit is in real space
347. /// so to find location in voxel space divide by voxelRes</returns>
348. private static Vector3 CompensateBestToolPlacement(
349.     int targetId,
350.     IHeightMap compMap,
351.     Vector2i centre,
352.     Func<float, float> toolMap,
353.     float toolRadius,
354.     out float bestCost)
355. {
356.     var bestPos = BestToolPlacement(targetId, compMap, centre, toolMap, toolRadius,
357.                                     out bestCost);
358.
359.     if (bestCost <= 1e-5f)
360.         return bestPos;
361.
362.     // update compMap with found position if necessary o.w. return

```

```

363.         var bestHeight = bestPos.Z;
364.
365.         if (Math.Abs(bestCost) <= float.Epsilon)
366.             return new Vector3(bestPos.X, bestPos.Y, bestHeight);
367.
368.         var minX = (int)((bestPos.X - toolRadius) / compMap.VoxelSize);
369.         var maxX = (int)((bestPos.X + toolRadius) / compMap.VoxelSize) + 1;
370.         var minY = (int)((bestPos.Y - toolRadius) / compMap.VoxelSize);
371.         var maxY = (int)((bestPos.Y + toolRadius) / compMap.VoxelSize) + 1;
372.
373.         // truncate
374.         minX = minX < 0 ? 0 : minX;
375.         maxX = maxX > compMap.Resolution.X ? compMap.Resolution.X : maxX;
376.         minY = minY < 0 ? 0 : minY;
377.         maxY = maxY > compMap.Resolution.Y ? compMap.Resolution.Y : maxY;
378.
379.         var radius2 = toolRadius * toolRadius;
380.
381.         for (var y = minY; y < maxY; ++y)
382.         {
383.             var yf = (y * compMap.VoxelSize - bestPos.Y);
384.
385.             for (int x = minX, linIdx = compMap.LinearIndex(minX, y); x < maxX; ++x,
++linIdx)
386.             {
387.                 var xf = (x * compMap.VoxelSize - bestPos.X);
388.                 var dist = xf * xf + yf * yf;
389.
390.                 if (dist > radius2)
391.                     continue;
392.
393.                 var inPlaneIdx = linIdx;
394.                 var lookup = compMap.Field[inPlaneIdx];
395.                 var height = toolMap(dist) + bestHeight;
396.
397.                 if (float.IsNaN(lookup) || float.IsNaN(height) || lookup > height)
398.                     continue;
399.
400.                 lookup = height;
401.
402.                 compMap.Field[inPlaneIdx] = lookup;
403.             }
404.         }
405.
406.         return new Vector3(bestPos.X, bestPos.Y, bestHeight);
407.     }
408.
409.     /// <summary>
410.     /// Generate best tool positions
411.     /// </summary>
412.     private IEnumerable<BestToolPosition> ToolPositionsToCompensate(
413.         IEnumerable<BestToolPosition> order, IHeightMap heightMap, IReadOnlyList<float>
targets)
414.     {
415.         var toolPositionsToCompensate = new List<BestToolPosition>();
416.
417.         foreach (var linIdx in order)
418.         {
419.             var comp = heightMap.Field[linIdx.Id];
420.
421.             // check if point is already satisfied
422.             if (float.IsNaN(comp) || comp > targets[linIdx.Id])
423.                 continue;
424.
425.             // if not select best possible tool comp for this location
426.             heightMap.PositionInVoxel(linIdx.Id, out var xPos, out var yPos);
427.

```

```

428.         var bestPos = CompensateBestToolPlacement(
429.             linIdx.Id,
430.             heightMap,
431.             new Vector2i(xPos, yPos),
432.             _toolShapeProvider.GetSqrDistFromHeadCenterToHeadSurface,
433.             _toolShapeProvider.BaseRadius,
434.             out var bestCost);
435.
436.         if (bestPos.Z <= 0 ||
437.             bestCost <= 0)
438.             continue;
439.
440.         toolPositionsToCompensate.Add(new BestToolPosition(0.0f, -1, bestPos));
441.     }
442.
443.     return toolPositionsToCompensate;
444. }
445.
446.     /// <summary>
447.     /// New ordering, sort on cost irregardless of neighbour tooling
448.     /// </summary>
449.     private IEnumerable<BestToolPosition> OrderPositionsByCost(IHeightMapReadOnly
heightMap, bool parallel)
450.     {
451.         if (parallel)
452.         {
453.             var order = Enumerable.Range(0, heightMap.Field.Count)
454.                 .AsParallel()
455.                 .Where(i => !float.IsNaN(heightMap.Field[i]))
456.                 .Select(i =>
457.                 {
458.                     heightMap.PositionInVoxel(i, out var x, out var y);
459.                     var centre = new Vector2i(x, y);
460.                     var bestPos = BestToolPlacement(i, heightMap, centre,
_toolShapeProvider.GetSqrDistFromHeadCenterToHeadSurface,
461. _toolShapeProvider.BaseRadius, out var cost);
462.                     return new BestToolPosition { BestCost = -cost, Id = i, BestPos =
bestPos };
463.                 })
464.                 .Where(bdp => bdp.BestCost < 0)
465.                 .OrderBy(bdp => bdp.BestCost)
466.                 .ToArray();
467.
468.             return order;
469.         }
470.         else
471.         {
472.             var order = Enumerable.Range(0, heightMap.Field.Count)
473.                 .Where(i => !float.IsNaN(heightMap.Field[i]))
474.                 .Select(i =>
475.                 {
476.                     heightMap.PositionInVoxel(i, out var x, out var y);
477.                     var centre = new Vector2i(x, y);
478.                     var bestPos = BestToolPlacement(i, heightMap, centre,
_toolShapeProvider.GetSqrDistFromHeadCenterToHeadSurface,
479. _toolShapeProvider.BaseRadius, out var cost);
480.                     return new BestToolPosition { BestCost = -cost, Id = i, BestPos =
bestPos };
481.                 })
482.                 .Where(bdp => bdp.BestCost < 0)
483.                 .OrderBy(bdp => bdp.BestCost)
484.                 .ToArray();
485.
486.             return order;
487.         }
488.     }
489.

```

```

490.     /// <summary>
491.     /// New ordering, sort on cost irregardless of neighbour tooling
492.     /// </summary>
493.     public IEnumerable<BestToolPosition>
        OrderPositionsByCostForChunk(IHeightMapReadOnly heightMap, IEnumerable<int> chunk)
494.     {
495.         var order = chunk
496.             .Where(i => !float.IsNaN(heightMap.Field[i]))
497.             .Select(i =>
498.             {
499.                 heightMap.PositionInVoxel(i, out var x, out var y);
500.                 var centre = new Vector2i(x, y);
501.                 var bestPos = BestToolPlacement(i, heightMap, centre,
                    _toolShapeProvider.GetSqrDistFromHeadCenterToHeadSurface,
502.                    _toolShapeProvider.BaseRadius, out var cost);
503.                 return new BestToolPosition { BestCost = -cost, Id = i, BestPos =
                    bestPos };
504.             })
505.             .Where(bdp => bdp.BestCost < 0)
506.             .OrderBy(bdp => bdp.BestCost)
507.             .ToArray();
508.
509.         return order;
510.     }
511.
512.     /// <summary>
513.     /// Builds tool compensation
514.     /// </summary>
515.     public IEnumerable<BestToolPosition> BuildToolCompensation(IHeightMapReadOnly
        heightMap, bool parallel)
516.     {
517.         // we adjust comp map hold a voxel res up
518.         // also need to store the original max heights for comparisons
519.         var targets = heightMap.Field.ToArray();
520.
521.         var order = OrderPositionsByCost(heightMap, parallel).ToArray();
522.
523.         var writableHeightMap = (IHeightMap)heightMap;
524.
525.         var toolPositionsToCompensate = ToolPositionsToCompensate(order,
            writableHeightMap, targets);
526.
527.         return toolPositionsToCompensate;
528.     }
529.
530.     /// <summary>
531.     /// Builds tool compensation
532.     /// </summary>
533.     public IEnumerable<BestToolPosition>
        BuildToolCompensationForChunks(IHeightMapReadOnly heightMap,
        IEnumerable<IEnumerable<BestToolPosition>> chunks)
534.     {
535.         // we adjust comp map hold a voxel res up
536.         // also need to store the original max heights for comparisons
537.         var targets = heightMap.Field.ToArray();
538.
539.         var aggregatedChunk = chunks
540.             .SelectMany(bdp => bdp)
541.             .OrderBy(bdp => bdp.BestCost);
542.
543.         var writableHeightMap = (IHeightMap)heightMap;
544.
545.         var toolPositionsToCompensate = ToolPositionsToCompensate(aggregatedChunk,
            writableHeightMap, targets);
546.
547.         return toolPositionsToCompensate;
548.     }

```

					ІАЛЦ.466500.009 Д6	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		14

```

549. }

1. // Types.cs
2. // Tool compensation. Distributed system.
3. // Bachelor thesis
4. // Oleksii Krutko. IO-z91
5. // 29.05.2023
6.
7. namespace ToolCompensation;
8.
9. public struct Vector2i
10. {
11.     public readonly int X;
12.     public readonly int Y;
13.
14.     public Vector2i(int x, int y)
15.     {
16.         X = x;
17.         Y = y;
18.     }
19. }
20.
21. public struct Vector3i
22. {
23.     public readonly int X;
24.     public readonly int Y;
25.     public readonly int Z;
26.
27.     public Vector3i(int x, int y, int z)
28.     {
29.         X = x;
30.         Y = y;
31.         Z = z;
32.     }
33. }

1. // ClientCalc.cs
2. // Tool compensation. Distributed system.
3. // Bachelor thesis
4. // Oleksii Krutko. IO-z91
5. // 29.05.2023
6.
7. using System.Numerics;
8. using ClientServerDataStructures;
9. using ToolCompensation;
10. using Utils;
11.
12. namespace Client;
13.
14. public class ClientCalc
15. {
16.     private readonly float _voxelSize;
17.     private readonly float _toolRadius;
18.     private readonly IHeightMapReadOnly _heightMap;
19.     private readonly IEnumerable<IServerInfo> _serverInfos;
20.     private readonly SynchronizedConsoleOutput _output;
21.
22.     public ClientCalc(float voxelSize, float toolRadius, IHeightMapReadOnly heightMap,
23.         IEnumerable<IServerInfo> serverInfos, SynchronizedConsoleOutput output)
24.     {
25.         _voxelSize = voxelSize;
26.         _toolRadius = toolRadius;
27.         _heightMap = heightMap;
28.         _serverInfos = serverInfos;
29.         _output = output;
30.     }
31.

```

```

32.     private IEnumerable<BestToolPosition> Calc()
33.     {
34.         var stream = new MemoryStream();
35.         var writer = new BinaryWriter(stream);
36.
37.         writer.Write(_voxelSize);
38.         writer.Write(_toolRadius);
39.
40.         _heightMap.WriteToStream(writer);
41.
42.         var threadsCount = _serverInfos.Sum(serverInfo => serverInfo.ThreadsCount);
43.
44.         var chunkSize = (_heightMap.Field.Count % threadsCount == 0)
45.             ? _heightMap.Field.Count / threadsCount
46.             : (_heightMap.Field.Count / threadsCount) + 1 + _heightMap.Field.Count %
threadsCount/16;
47.         var chunks = Enumerable
48.             .Range(0, _heightMap.Field.Count)
49.             .Chunk(chunkSize)
50.             .ToList();
51.
52.         var connections = new List<IServerConnection>();
53.         var positionsChunks = new List<List<BestToolPosition>>();
54.         var iChunk = 0;
55.         foreach (var serverInfo in _serverInfos)
56.         {
57.             var streamPerNode = new MemoryStream();
58.             var writerPerNode = new BinaryWriter(streamPerNode);
59.             writerPerNode.Write(stream.ToArray());
60.             writerPerNode.Write(serverInfo.ThreadsCount);
61.             var chunksForNode = chunks
62.                 .GetRange(iChunk, serverInfo.ThreadsCount)
63.                 .ToArray();
64.
65.             foreach (var chunk in chunksForNode)
66.             {
67.                 writerPerNode.Write(chunk.Length);
68.                 foreach (var item in chunk)
69.                 {
70.                     writerPerNode.Write(item);
71.                 }
72.             }
73.
74.             var conn = new ServerConnectionSocket(serverInfo, _output, "1");
75.             connections.Add(conn);
76.             if (!conn.Connect())
77.             {
78.                 _output.WriteLine($"Could not connect to {serverInfo.Name} -
{serverInfo.IP}: {serverInfo.Port}");
79.                 return Enumerable.Empty<BestToolPosition>();
80.             }
81.
82.             streamPerNode.Write(new byte[] { 0xBA, 0xBA, 0xBA, 0xBA, 0xBA, 0xBA, 0xBA, 0xBA });
83.
84.             conn.Send(streamPerNode.ToArray());
85.
86.             iChunk += serverInfo.ThreadsCount;
87.         }
88.
89.         foreach (var conn in connections)
90.         {
91.             conn.Receive();
92.
93.             var dataStream = new MemoryStream(conn.ReceivedData);
94.             dataStream.Position = 0;
95.             var binaryReader = new BinaryReader(dataStream);
96.

```

```

97.         var positions = new List<BestToolPosition>();
98.         var positionsCount = binaryReader.ReadInt32();
99.         for (var i = 0; i < positionsCount; i++)
100.        {
101.            var bestCost = binaryReader.ReadSingle();
102.            var id = binaryReader.ReadInt32();
103.            var x = binaryReader.ReadSingle();
104.            var y = binaryReader.ReadSingle();
105.            var z = binaryReader.ReadSingle();
106.
107.            var bestToolPosition = new BestToolPosition(bestCost, id, new
Vector3(x, y, z));
108.            positions.Add(bestToolPosition);
109.        }
110.        positionsChunks.Add(positions);
111.    }
112.
113.    foreach (var conn in connections)
114.    {
115.        conn.Disconnect();
116.    }
117.
118.    var sphericalTool = new SphericalTool(_toolRadius, 20.0f);
119.    var toolCompensationBuilder =
120.        (IToolCompensationBuilder)new
ThreeZeroAxisToolCompensationBuilder(sphericalTool);
121.
122.    var positionsFinal =
toolCompensationBuilder.BuildToolCompensationForChunks(_heightMap, positionsChunks);
123.
124.    return positionsFinal;
125. }
126.
127. public async Task<IEnumerable<BestToolPosition>> CalcAsync()
128. {
129.     return await Task.Run(Calc);
130. }
131. }

```

```

1. // IServerConnection.cs
2. // Tool compensation. Distributed system.
3. // Bachelor thesis
4. // Oleksii Krutko. IO-z91
5. // 29.05.2023
6.
7. namespace Client;
8.
9. /// <summary>Connection with a server</summary>
10. public interface IServerConnection
11. {
12.     /// <summary>The response from the remote device</summary>
13.     public byte[] ReceivedData { get; }
14.
15.     /// <summary>Number of threads available on server</summary>
16.     public int ThreadsCount { get; }
17.
18.     /// <summary>Connect to server</summary>
19.     public bool Connect();
20.
21.     /// <summary>Disconnect from server</summary>
22.     public void Disconnect();
23.
24.     /// <summary>Receive data from server</summary>
25.     public void Receive();
26.
27.     /// <summary>Send data to server</summary>
28.     public void Send(byte[] data);

```

Змн.	Арк.	№ докум.	Підпис	Дата

ІАЛЦ.466500.009 Д6

Арк.

17

```

29.
30.     /// <summary>Send done event</summary>
31.     public ManualResetEvent SendDoneEvent { get; }
32.
33.     /// <summary>Receive done event</summary>
34.     public ManualResetEvent ReceiveDoneEvent { get; }
35. }

1.  // ServerConnectionSocket.cs
2.  // Tool compensation. Distributed system.
3.  // Bachelor thesis
4.  // Oleksii Krutko. IO-z91
5.  // 29.05.2023
6.
7.  using System.Net;
8.  using System.Net.Sockets;
9.  using ClientServerDataStructures;
10. using Utils;
11.
12. namespace Client;
13.
14. /// <summary>Connection with a server using Sockets</summary>
15. public class ServerConnectionSocket : IServerConnection
16. {
17.     // Log output
18.     private readonly SynchronizedConsoleOutput _output;
19.
20.     // Client ID
21.     private readonly string _clientId;
22.
23.     // Server ID
24.     private readonly string _serverId;
25.
26.     private readonly IPEndPoint _ipEndPoint;
27.     private readonly Socket _client;
28.
29.     // ManualResetEvent instances signal completion.
30.     private readonly ManualResetEvent _connectDone = new(false);
31.     private readonly ManualResetEvent _sendDone = new(false);
32.     private readonly ManualResetEvent _receiveDone = new(false);
33.
34.     /// <inheritdoc />
35.     public ManualResetEvent SendDoneEvent => _sendDone;
36.
37.     /// <inheritdoc />
38.     public ManualResetEvent ReceiveDoneEvent => _receiveDone;
39.
40.     /// <inheritdoc />
41.     public byte[] ReceivedData { get; private set; }
42.
43.     /// <inheritdoc />
44.     public int ThreadsCount { get; }
45.
46.     /// <summary>Constructor</summary>
47.     public ServerConnectionSocket(IServerInfo serverInfo, SynchronizedConsoleOutput
output, string clientId)
48.     {
49.         _output = output;
50.         _clientId = clientId;
51.         _serverId = serverInfo.Port.ToString();
52.         ThreadsCount = serverInfo.ThreadsCount;
53.         ReceivedData = Array.Empty<byte>();
54.
55.         // Set server's endpoint for the socket.
56.         var ipAddress = serverInfo.IP;
57.         _ipEndPoint = new IPEndPoint(ipAddress, serverInfo.Port);
58.

```

```

59.         // Create a TCP/IP socket.
60.         _client = new Socket(ipAddress.AddressFamily,
61.             SocketType.Stream, ProtocolType.Tcp);
62.     }
63.
64.     /// <inheritdoc />
65.     public bool Connect()
66.     {
67.         _output.WriteLine($"Connecting to server {_serverId} ...");
68.         _client.BeginConnect(_ipEndPoint, ConnectCallback, _client);
69.         return _connectDone.WaitOne(1000);
70.     }
71.
72.     /// <inheritdoc />
73.     public void Disconnect()
74.     {
75.         _client.Shutdown(SocketShutdown.Both);
76.         _client.Close();
77.     }
78.
79.     private void ConnectCallback(IAsyncResult ar) {
80.         try {
81.             // Retrieve the socket from the state object.
82.             var client = (Socket) ar.AsyncState!;
83.
84.             // Complete the connection.
85.             client.EndConnect(ar);
86.
87.             _output.WriteLine($"Client {_clientId} connected to server {_serverId}
{client.RemoteEndPoint}");
88.
89.             // Signal that the connection has been made.
90.             _connectDone.Set();
91.         } catch (Exception e) {
92.             _output.WriteLine($"Client {_clientId} could not connect to server
{_serverId}. Error {e}");
93.         }
94.     }
95.
96.     /// <inheritdoc />
97.     public void Receive() {
98.         try
99.         {
100.             _output.WriteLine($"Receiving data from server {_serverId}...");
101.
102.             ReceivedData = Array.Empty<byte>();
103.             _receiveDone.Reset();
104.
105.             var stateObject = new StateObject
106.             {
107.                 client = _client,
108.                 bufferStream = new MemoryStream()
109.             };
110.
111.             // Begin receiving the data from the remote device.
112.             _client.BeginReceive( stateObject.buffer, 0, CommonConstants.PacketSize,
0,
113.                 ReceiveCallback, stateObject);
114.
115.             _receiveDone.WaitOne();
116.             if (stateObject.bufferStream.Length > 1)
117.             {
118.                 //ALL of the data has been read, so displays it to the console
119.                 ReceivedData = stateObject.bufferStream.ToArray();
120.             }
121.
122.             // Write the response to the console.

```

```

123.         _output.WriteLine($"Data received form serve {_serverId}.
           {ReceivedData.Length} bytes read");
124.     } catch (Exception e) {
125.         _output.WriteLine(e.ToString());
126.     }
127. }
128.
129. private void ReceiveCallback( IAsyncResult ar ) {
130.     try
131.     {
132.         // Retrieve the state object and the client socket
133.         // from the asynchronous state object.
134.         var stateObject = ar.AsyncState as StateObject;
135.
136.         if (stateObject?.bufferStream == null)
137.             throw new Exception("No state object!");
138.
139.         // Read data from the remote device.
140.         var bytesRead = stateObject.client!.EndReceive(ar);
141.
142.         if (bytesRead > 0)
143.         {
144.             // There might be more data, so store the data received so far.
145.             stateObject.bufferStream.Write(stateObject.buffer, 0, bytesRead);
146.
147.             if (stateObject.bufferStream.Length > 8)
148.             {
149.                 stateObject.bufferStream.Position = stateObject.bufferStream.Length
150.                 - 8;
151.
152.                 var eof = new byte[8];
153.                 var readBytes = stateObject.bufferStream.Read(eof, 0, 8);
154.
155.                 if (readBytes == 8 && eof.SequenceEqual(new byte[]
156.                 {0xBA,0xBA,0xBA,0xBA,0xBA,0xBA,0xBA,0xBA}))
157.                 {
158.                     _receiveDone.Set();
159.                     return;
160.                 }
161.
162.                 // Get the rest of the data.
163.                 stateObject.client.BeginReceive(stateObject.buffer, 0,
164.                 CommonConstants.PacketSize, 0,
165.                 ReceiveCallback, stateObject);
166.             }
167.             else{
168.                 _output.WriteLine($"Client disconnected");
169.             }
170.             //_receiveDone.Set();
171.         }
172.         catch (ObjectDisposedException)
173.         {
174.             _output.WriteLine($"Client {_clientId} closed connection {_serverId}.");
175.         }
176.         catch (Exception e) {
177.             _output.WriteLine($"Client {_clientId} could not receive data from server
178.             {_serverId}. Error {e}");
179.         }
180.     }
181.
182.     /// <inheritdoc />
183.     public void Send(byte[] data)
184.     {
185.         _output.WriteLine($"Sending data to server {_serverId}...");
186.
187.         // Begin sending the data to the remote device.
188.         _client.BeginSend(data, 0, data.Length, 0,

```

```

185.         SendCallback, _client);
186.
187.         _sendDone.WaitOne();
188.
189.         _output.WriteLine($"Data has been sent to server {_serverId}");
190.     }
191.
192.     private void SendCallback(IAsyncResult ar) {
193.         try {
194.             // Get socket from the state object.
195.             var client = (Socket) ar.AsyncState!;
196.
197.             // Complete sending the data to server.
198.             var bytesSent = client.EndSend(ar);
199.             _output.WriteLine($"Sent {bytesSent} bytes to server {_serverId}.");
200.
201.             // Signal that all bytes have been sent.
202.             _sendDone.Set();
203.         } catch (Exception e) {
204.             _output.WriteLine($"Client {_clientId} could not send data to server
205. {_serverId}. Error {e}");
206.         }
207.     }

```

```

1. // CalcServer.cs
2. // Tool compensation. Distributed system.
3. // Bachelor thesis
4. // Oleksii Krutko. IO-z91
5. // 29.05.2023
6.
7. using System.Collections.Concurrent;
8. using ClientServerDataStructures;
9. using ToolCompensation;
10. using Utils;
11.
12. namespace Server;
13.
14. public class CalcServer
15. {
16.     // Log output
17.     private readonly SynchronizedConsoleOutput _output;
18.
19.     // Active connection
20.     private readonly IClientConnection _client;
21.
22.     private float _voxelSize;
23.     private float _toolRadius;
24.     private IHeightMapReadOnly? _heightMap;
25.     private List<List<int>>>? _chunks;
26.
27.     private SphericalTool? _sphericalTool;
28.     private ThreeZeroAxisToolCompensationBuilder? _toolCompensationBuilder;
29.
30.     // Number of threads. 3 minimum
31.     private readonly int _nThreads;
32.
33.     // Output
34.     private readonly ConcurrentBag<List<BestToolPosition>> _result;
35.
36.     // Event Input
37.     private readonly ManualResetEvent _inputReadyEvent;
38.     // Event e
39.     private readonly List<ManualResetEvent> _eResultReady;
40.
41.     /// <summary>Constructor</summary>

```

```

42.     public CalcServer(IClientConnection clientConnection, SynchronizedConsoleOutput
output,
43.         ISystemInfoProvider sysInfo, int threadsCount)
44.     {
45.         _output = output ?? throw new ArgumentNullException(nameof(output));
46.
47.         _output.WriteLine("Initializing server...");
48.
49.         if (sysInfo == null)
50.             throw new ArgumentNullException(nameof(sysInfo));
51.
52.         var cpuInfos = sysInfo.GetCpuInfos().ToArray();
53.
54.         if(!cpuInfos.Any())
55.             throw new Exception("No CPU found");
56.
57.         foreach (var cpuInfo in sysInfo.GetCpuInfos())
58.         {
59.             output.WriteLine(cpuInfo.ToString());
60.         }
61.
62.         _nThreads = threadsCount;
63.
64.         _inputReadyEvent = new ManualResetEvent(false);
65.
66.         _eResultReady = new List<ManualResetEvent>();
67.         _result = new ConcurrentBag<List<BestToolPosition>>();
68.
69.         for (var i = 0; i < _nThreads; i++)
70.         {
71.             _eResultReady.Add(new ManualResetEvent(false));
72.         }
73.
74.         _client = clientConnection;
75.     }
76.
77.
78.     /// <summary>Starts calculation server</summary>
79.     public void Start()
80.     {
81.         _client.Start(Run);
82.     }
83.
84.     private void Run()
85.     {
86.         _inputReadyEvent.Reset();
87.         _result.Clear();
88.
89.         for (var i = 0; i < _nThreads; i++)
90.         {
91.             _eResultReady[i].Reset();
92.         }
93.
94.         var threadPool = new List<Thread>
95.         {
96.             new(() => ServerCommunicationTask(_client)) {Name = "Communication"}
97.         };
98.
99.         for (var i = 0; i < _nThreads; i++)
100.        {
101.            var id = i;
102.            threadPool.Add(new Thread(() => ServerCalculationTask(id)) {Name =
103.                $"Calculation {id}"});
104.        }
105.
106.        foreach (var thread in threadPool)

```

```

107.         thread.Start();
108.     }
109.
110.     _output.WriteLine("Waiting for all threads");
111.
112.     foreach (var thread in threadPool)
113.     {
114.         thread.Join();
115.     }
116. }
117.
118. private void ServerCommunicationTask(IClientConnection client)
119. {
120.     _output.WriteLine($"Started");
121.
122.     client.Receive();
123.
124.     InitInputData(client.ReceivedData);
125.
126.     _inputReadyEvent.Set();
127.
128.     // Wait for all parts ready
129.     WaitHandle.WaitAll(_eResultReady.Cast<WaitHandle>().ToArray());
130.
131.     // Output result
132.     var combined = _result
133.         .SelectMany(btp => btp)
134.         .OrderBy(bdp => bdp.BestCost)
135.         .ToArray();
136.
137.     var stream = new MemoryStream();
138.     stream.Position = 0;
139.     var writer = new BinaryWriter(stream);
140.     writer.Write(combined.Length);
141.     foreach (var bestToolPosition in combined)
142.     {
143.         writer.Write(bestToolPosition.BestCost);
144.         writer.Write(bestToolPosition.Id);
145.         writer.Write(bestToolPosition.BestPos.X);
146.         writer.Write(bestToolPosition.BestPos.Y);
147.         writer.Write(bestToolPosition.BestPos.Z);
148.     }
149.
150.     writer.Write(new byte[] { 0xBA, 0xBA, 0xBA, 0xBA, 0xBA, 0xBA, 0xBA, 0xBA });
151.
152.     client.Send(stream.ToArray());
153.
154.     _output.WriteLine($"Finished");
155. }
156.
157. private void ServerCalculationTask(int id)
158. {
159.     _output.WriteLine($"Started");
160.
161.     _inputReadyEvent.WaitOne();
162.
163.     CalcExpressionForRange(id);
164.
165.     _eResultReady[id].Set();
166.
167.     // Output result
168.     _output.WriteLine($"Finished");
169. }
170.
171. private void InitInputData(byte[] data)
172. {
173.     var dataStream = new MemoryStream(data);

```

```

174.         dataStream.Position = 0;
175.         var binaryReader = new BinaryReader(dataStream);
176.
177.         _voxelSize = binaryReader.ReadSingle();
178.         _toolRadius = binaryReader.ReadSingle();
179.         _heightMap = HeightMap.ReadFromStream(binaryReader);
180.
181.         var chunksCount = binaryReader.ReadInt32();
182.         _chunks = new List<List<int>>();
183.         for (var i = 0; i < chunksCount; i++)
184.         {
185.             var chunk = new List<int>();
186.             var itemsCount = binaryReader.ReadInt32();
187.             for (var j = 0; j < itemsCount; j++)
188.             {
189.                 chunk.Add(binaryReader.ReadInt32());
190.             }
191.             _chunks.Add(chunk);
192.         }
193.
194.         _sphericalTool = new SphericalTool(_toolRadius, 20.0f);
195.         _toolCompensationBuilder =
196.             new ThreeZeroAxisToolCompensationBuilder(_sphericalTool);
197.     }
198.
199.     private void CalcExpressionForRange(int part)
200.     {
201.         _output.WriteLine($"Calculation part {part + 1}");
202.
203.         if (_heightMap == null || _chunks == null || _toolCompensationBuilder == null)
204.         {
205.             _output.WriteLine($"Data is not ready. Aborting part {part + 1}");
206.             return;
207.         }
208.     }
209.
210.     _result.Add(_toolCompensationBuilder.OrderPositionsByCostForChunk(_heightMap,
211.         _chunks[part]).ToList());
212. }

```

```

1. // ClientConnectionSocket.cs
2. // Tool compensation. Distributed system.
3. // Bachelor thesis
4. // Oleksii Krutko. IO-z91
5. // 29.05.2023
6.
7. using System.Net;
8. using System.Net.Sockets;
9. using ClientServerDataStructures;
10. using Utils;
11.
12. namespace Server;
13.
14. /// <summary>Connection with a client using Sockets.</summary>
15. public class ClientConnectionSocket: IClientConnection
16. {
17.     // Log output
18.     private readonly SynchronizedConsoleOutput _output;
19.
20.     private readonly IPEndPoint _ipEndPoint;
21.     private readonly Socket _listener;
22.     private Socket? _handler;
23.
24.     // ManualResetEvent instances signal completion.
25.     private readonly ManualResetEvent _connectDone = new(false);
26.     private readonly ManualResetEvent _sendDone = new(false);

```

```

27.     private readonly ManualResetEvent _receiveDone = new(false);
28.
29.     private readonly ManualResetEvent _calculationDone =
30.         new ManualResetEvent(false);
31.
32.     /// <inheritdoc />
33.     public byte[] ReceivedData { get; private set; }
34.
35.     public ClientConnectionSocket(int port, SynchronizedConsoleOutput output)
36.     {
37.         _output = output;
38.
39.         var ipHostInfo = Dns.GetHostEntry(Dns.GetHostName());
40.         var ipAddress = ipHostInfo.AddressList[3];
41.         _ipEndPoint = new IPEndPoint(ipAddress, port);
42.
43.         ReceivedData = Array.Empty<byte>();
44.
45.         // Create a TCP/IP socket.
46.         _listener = new Socket(AddressFamily.InterNetwork,
47.             SocketType.Stream, ProtocolType.Tcp );
48.
49.         _handler = null;
50.     }
51.
52.     /// <inheritdoc />
53.     public void Start(Action run)
54.     {
55.         _output.WriteLine("Starting calculation server...");
56.
57.         // Bind the socket to the local endpoint and listen for incoming connections.
58.         try
59.         {
60.             _listener.Bind(_ipEndPoint);
61.             _listener.Listen(100);
62.
63.             while (true)
64.             {
65.                 // Set the event to non-signaled state.
66.                 _calculationDone.Reset();
67.                 _connectDone.Reset();
68.
69.                 // Start an asynchronous socket to listen for connections.
70.                 _output.WriteLine($"Waiting for client ...");
71.                 _listener.BeginAccept(AcceptCallback, _listener);
72.
73.                 // Wait until a connection is made before continuing.
74.                 _connectDone.WaitOne();
75.
76.                 run();
77.             }
78.
79.         } catch (Exception e) {
80.             _output.WriteLine($"Server could not initialize. Error {e}");
81.         }
82.
83.         _output.WriteLine("Calculation server finished...");
84.     }
85.
86.     private void AcceptCallback(IAsyncResult ar)
87.     {
88.         // Get the socket that handles the client request.
89.         if (ar.AsyncState is not Socket listener)
90.             throw new Exception("No valid socket while connecting client!");
91.
92.         _handler = listener.EndAccept(ar);
93.

```

```

94.         _output.WriteLine($"Client connected to server {_handler.RemoteEndPoint}");
95.
96.         _output.WriteLine("Client connected");
97.
98.         _connectDone.Set();
99.     }
100.
101.     /// <inheritdoc />
102.     public void Receive()
103.     {
104.         _output.WriteLine("Receiving data from client...");
105.
106.         ReceivedData = Array.Empty<byte>();
107.         _receiveDone.Reset();
108.
109.         // Create the state object.
110.         var state = new StateObject
111.         {
112.             client = _handler,
113.             bufferStream = new MemoryStream()
114.         };
115.
116.         _handler?.BeginReceive( state.buffer, 0, CommonConstants.PacketSize, 0,
117.             ReceiveCallback, state);
118.
119.         _receiveDone.WaitOne();
120.         if (state.bufferStream.Length > 1)
121.         {
122.             //ALL of the data has been read, so displays it to the console
123.             ReceivedData = state.bufferStream.ToArray();
124.         }
125.
126.         _output.WriteLine($"Data has been received from client. {ReceivedData.Length}
bytes read");
127.     }
128.
129.     private void ReceiveCallback(IAsyncResult ar)
130.     {
131.         try
132.         {
133.             // Retrieve the state object and the client socket
134.             // from the asynchronous state object.
135.             var stateObject = ar.AsyncState as StateObject;
136.
137.             if (stateObject?.bufferStream == null)
138.                 throw new Exception("No state object!");
139.
140.             // Read data from the remote device.
141.             var bytesRead = stateObject.client!.EndReceive(ar);
142.
143.             if (bytesRead > 0)
144.             {
145.                 // There might be more data, so store the data received so far.
146.                 stateObject.bufferStream.Write(stateObject.buffer, 0, bytesRead);
147.
148.                 if (stateObject.bufferStream.Length > 8)
149.                 {
150.                     stateObject.bufferStream.Position = stateObject.bufferStream.Length
- 8;
151.
152.                     var eof = new byte[8];
153.                     var readBytes = stateObject.bufferStream.Read(eof, 0, 8);
154.
155.                     if (readBytes == 8 && eof.SequenceEqual(new byte[]
{0xBA, 0xBA, 0xBA, 0xBA, 0xBA, 0xBA, 0xBA, 0xBA}))
156.                     {
157.                         _receiveDone.Set();
158.                         return;
159.                     }
160.                 }
161.             }
162.         }
163.         catch { }
164.     }

```

```

158.         }
159.     }
160.
161.     // Get the rest of the data.
162.     stateObject.client.BeginReceive(stateObject.buffer, 0, CommonConstants.Pa
cketSize, 0,
163.         ReceiveCallback, stateObject);
164.     }
165.     else{
166.         _output.WriteLine($"Client disconnected");
167.     }
168. } catch (Exception e) {
169.     _output.WriteLine($"Server could not receive data from client. Error
{e}");
170. }
171. }
172.
173. /// <inheritdoc />
174. public void Send( byte[] data)
175. {
176.     _output.WriteLine("Sending data to client...");
177.
178.     // Begin sending the data to the remote device.
179.     _handler?.BeginSend(data, 0, data.Length, 0, SendCallback, _handler);
180.
181.     _sendDone.WaitOne();
182. }
183.
184. private void SendCallback(IAsyncResult ar)
185. {
186.     try
187.     {
188.         // Retrieve the socket from the state object.
189.         if (ar.AsyncState is not Socket handler)
190.             throw new Exception("No valid socket while sending input data to
client!");
191.
192.         // Complete sending the data to the remote device.
193.         var bytesSent = handler.EndSend(ar);
194.         _output.WriteLine($"Sent {bytesSent} bytes to client.");
195.
196.         _sendDone.Set();
197.     }
198.     catch (Exception e)
199.     {
200.         _output.WriteLine($"Input data could not be sent to a client. Error {e}");
201.     }
202. }
203.
204. }

```

```

1. // CommandLineOptions.cs
2. // Tool compensation. Distributed system.
3. // Bachelor thesis
4. // Oleksii Krutko. IO-z91
5. // 29.05.2023
6.
7. using CommandLine;
8. using CommandLine.Text;
9.
10. namespace Server;
11.
12. internal class CommandLineOptions
13. {
14.     [Option('p', "Port", Required = true, HelpText = "Specifies the port that is being
listening")]
15.     public int Port { get; private init; }

```

```

16.
17.     [Option('t', "Threads", Required = true, HelpText = "Specifies the of threads to use
    ")]
18.     public int Threads { get; private init; }
19.
20.     [Usage(ApplicationAlias = "ToolCompensationServer")]
21.     public static IEnumerable<Example> Examples =>
22.         new List<Example>() {
23.             new("Start tool compensation server listening on port", new
24. CommandLineOptions { Port = 10001 }),
25.             new("Start tool compensation server using specific number of threads", new
26. CommandLineOptions { Threads = 4 })
27.         };
28. internal enum ExitCode
29. {
30.     Success = 0,
31.     InvalidArguments = 1,
32.     UnhandledException = 2,
33. };

1. // Program.cs
2. // Tool compensation. Distributed system.
3. // Bachelor thesis
4. // Oleksii Krutko. IO-z91
5. // 29.05.2023
6.
7. using Server;
8. using Utils;
9. using CommandLine;
10.
11. var options = new CommandLineOptions();
12. var result = Parser.Default.ParseArguments<CommandLineOptions>(args)
13.     .WithParsed(o => options = o);
14. if (result.Tag == ParserResultType.NotParsed)
15. {
16.     Environment.ExitCode = (int) ExitCode.InvalidArguments;
17.     return;
18. }
19.
20. try
21. {
22.     var cpuInfoProvider = new SystemInfoProvider();
23.
24.     var output = new SynchronizedConsoleOutput(Console.Out);
25.
26.     var clientConnectionSocket = new ClientConnectionSocket(options.Port, output);
27.
28.     var calcServer = new CalcServer(clientConnectionSocket, new
29. SynchronizedConsoleOutput(Console.Out),
30.     cpuInfoProvider, options.Threads);
31.     calcServer.Start();
32. }
33. catch (Exception e)
34. {
35.     Console.WriteLine(e.Message);
36.     Environment.ExitCode = (int) ExitCode.UnhandledException;
37. }
38. Environment.ExitCode = (int) ExitCode.Success;

```