

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА РОЗРОБКА ЗАСОБІВ ОБРОБЛЕННЯ ДАНИХ
ДЛЯ АНАЛІЗУ РЕЗУЛЬТАТІВ ВИПРОБУВАНЬ ТЕХНІЧНИХ ОБ'ЄКТІВ
RESEARCH AND IMPLEMENTATION OF DATA PROCESSING TOOLS
FOR TESTING RESULTS OF TECHNICAL OBJECTS

Виконав: студент(ка) 2 курсу, групи КНТ-219м
Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Системи штучного інтелекту

Кугай А.Е.

(прізвище та ініціали)

Керівник Шитікова О.В.

(прізвище та ініціали)

Рецензент Терещенко Е.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 122 Комп'ютерні науки
 (код і найменування)
 Освітня програма (спеціалізація) Системи штучного інтелекту
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 20 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Кугай Альони Едуардівни

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та розробка засобів оброблення даних для аналізу результатів випробувань технічних об'єктів

Research and Implementation of Data Processing Tools for Testing Results of Technical Objects

керівник проєкту (роботи) Шитікова Олена Вікторівна, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “3” листопада 2020 року № 301

2. Строк подання студентом проєкту (роботи) 15 грудня 2020 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження.

2. Матеріали і методи. 3. Розробка архітектури програми. 4. Розробка програми.

5. Експлуатація, тестування та експериментальне дослідження програми.

6. Економіко-організаційна частина. 7. Охорона праці та безпека в надзвичайних ситуаціях.

8. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Шитікова О.В, доцент		
Економіко-організаційна частина	Пожуєва Т.О., професор		
Охорона праці та безпека в надзвичайних ситуаціях	Коробко О.В., ст. викладач		
Нормоконтролер	Андрєєв М.О., асистент		

7. Дата видачі завдання “ 4 ” вересня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей оброблення даних для аналізу результатів випробувань технічних об'єктів.	4–5 тижні	Розділ 2
4	Розробка архітектури програми з використанням LS-DYNA.	6 тиждень	Розділ 3
5	Розробка програми.	7–8 тижні	Розділ 4
6	Тестування та експериментальне дослідження засобів оброблення даних для аналізу результатів випробувань технічних об'єктів.	9 тиждень	Розділ 5
7	Розробка економіко-організаційної частини	10 тиждень	Розділ 6
8	Розробка заходів з охорони праці та безпеки в надзвичайних ситуаціях.	11 тиждень	Розділ 7
9	Оформлення пояснювальної записки та документів до неї.	12-13 тиждень	Додатки
10	Нормоконтроль та рецензування.	14–15 тижні	
11	Захист роботи.	16 тиждень	

Студент(ка)

_____ Кугай А.Е.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)

_____ Шитікова О.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
134 с., 6 табл., рис. 42., 2 дод., 30 джерел.

КЛАСТЕР, LS-DYNA, КРАШ-ТЕСТ, BOINC, СУПЕРКОМП'ЮТЕР,
PYTHON.

Об'єкт дослідження – програмні засоби для обробки даних аналізу результатів випробувань технічних об'єктів.

Предмет дослідження – комп'ютерні програми для обробки даних аналізу результатів випробувань технічних об'єктів.

Мета роботи – підвищення швидкості обчислень за рахунок створення засобу для оброблення даних та аналізу результатів випробувань технічних об'єктів.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, середовище моделювання LS-DYNA, мови програмування Python, платформа BOINC, 25 персональних комп'ютерів з процесором Core I5 під управлінням операційної системи Microsoft Windows 10 та Linux.

Результати. Розроблений засіб, за умови використання 25 робочих станцій, дозволяє проводити розрахунки на 65% швидше ніж на 1 робочій станції.

Висновки. Розроблений засіб дозволяє окремим користувачам легко і швидко взаємодіяти один з одним та виконувати розрахунки для проєкту. Таке рішення дозволяє проводити всі необхідні розрахунки на базі робочих станцій на виробництві.

Галузь використання – засіб може використовуватися в автомобілебудуванні та при проведенні масштабних обчислень.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 134 pages, 42 figures, 6 tables, 2 appendixes, 30 sources.

CLUSTER, LS-DYNA, CRASH TEST, BOINC, SUPERCOMPUTER, PYTHON.

Object of study is software for data processing of analysis of test results of technical objects.

Subject of study are is computer programs for data processing of analysis of test results of technical objects.

The purpose of the work is increasing the speed of calculations by creating a tool for data processing and analysis of test results of technical objects.

Materials, methods, and tools: structured and object-oriented programming, LS-DYNA modeling environment, Python programming languages, BOINC platform, 25 personal computers with Core I5 processor running Microsoft Windows 10 and Linux.

Results. The developed tool, provided the use of 25 workstations, allows calculations to be performed 65% faster than 1 workstation.

Conclusions. The developed tool allows individual users to easily and quickly interact with each other and perform calculations for the project. This solution allows you to perform all necessary calculations on the basis of workstations in production.

Field of use - the tool can be used in the automotive industry and in large-scale calculations.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Випробування технічних об'єктів.....	10
1.2 Засіб обробки інформації - кластер.....	11
1.3 Засіб обробки інформації - суперкомп'ютер	15
2 Матеріали і методи.....	17
2.1 Існуючі аналоги системи	17
2.1.1 Суперкомп'ютерний обчислювальний центр СКІТ – 1.....	17
2.2.2 Суперкомп'ютерний обчислювальний центр СКІТ – 2.....	18
2.2.3 Суперкомп'ютерний обчислювальний центр СКІТ – 3.....	18
2.2.4 Суперкомп'ютерний обчислювальний центр СКІТ – 4.....	19
2.2 Вибір апаратної та програмної частини для дослідження	20
3 Розробка архітектури програми.....	35
3.1 Запуск кластера	35
3.2 Проєктування в програмному середовищі LS DYNA.....	36
3.3 Зчитувач файлів.....	44
3.4 Формування завдання в Voinc.....	46
4 Розробка програми	47
4.1 Налаштування кластеру.....	47
4.2 Налаштування Voinc- сервера.....	48
4.3 Структура сценаріїв Python ¶.....	52
5 Експлуатація, тестування та експериментальне дослідження програми	56
5.1 Налаштування LS-DYNA	56
5.2 Налаштування Voinc-клієнта	58
6 Економіко-організаційна частина.....	60
6.1 Планування розробки програмного продукту.....	60

6.2	Визначення витрат на розробку програми	61
6.3	Розрахунок економічної ефективності програмного продукту.....	68
7	Охорона праці та безпека в надзвичайних ситуаціях.....	71
7.1	Аналіз потенційних небезпек.....	71
7.2	Заходи із забезпечення безпеки	72
7.3	Заходи з виробничої санітарії та гігієни праці.....	74
7.4	Заходи безпеки у надзвичайних ситуаціях.....	79
7.4.1	Заходи з пожежної безпеки	79
7.4.2	Заходи з цивільного захисту	81
	Висновок	85
	Перелік джерел посилання	86
	Додаток А Текст програми.....	89
	Додаток Б Слайди презентації	127

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
E	– Ethernet;
EuroNCAP	– The European New Car Assessment Programme;
Fe	– Fast Ethernet;
GE	– Gigabit Ethernet;
MPP	– Integrated Development Environment;
SMP	– Symmetric multiprocessing;
VIA	– Virtual Interface Architecture;
ПК	– персональний комп'ютер.

ВСТУП

Високопродуктивні обчислення грають все більшу роль при проведенні наукових досліджень, розробці нових видів промислової продукції і в соціальній сфері. Вони високо затребувані як інструмент для вирішення завдань квантової хімії, молекулярної біології, гідродинаміки та інших галузей наук, що вимагають проведення великої кількості ресурсоемних обчислювальних експериментів і обробки великих обсягів даних. З урахуванням виникнення нових обчислювальних завдань і необхідністю обробки зростаючих обсягів інформації, актуальним завданням є розробка і розвиток технологій для проведення ресурсномістких обчислень в рамках різних наукових і прикладних проєктів.

Однак виникає проблема в тому що подібні розрахунки потребують додаткових фінансових вкладів. Тому задача створення системи для проведення таких розрахунків є актуально.

Для даного проєкту та моделювання засобу оброблення даних для аналізу результатів випробувань технічних об'єктів було обрано завдання розрахунків краш тестів автомобілів.

Мета роботи полягає у підвищенні швидкості обчислень за рахунок створення засобу для оброблення даних та аналізу результатів випробувань технічних об'єктів

Об'єкт – процес обробки та аналізу даних.

Предмет дослідження – є спроектована інформаційна інфраструктура для оброблення даних та аналізу результатів випробувань технічних об'єктів

Відповідно до мети були поставлені наступні завдання:

- провести аналіз засобів для оброблення даних та аналізу результатів випробувань технічних об'єктів;
- провести аналіз існуючих програмних рішень;
- провести моделювання та розрахунок випробування.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Випробування технічних об'єктів

В автомобілебудуванні з кожним роком все більше уваги приділяється безпеці автомобілів. На сьогоднішній день практично всі моделі автомобілів, особливо популярних виробників, перш ніж надійти в продаж, проходять краш-тести. Краш-тест це випробування автомобілів в спеціально відтворених умовах аварій. Це штучне подоби аварії, коли уникнути її не вдалося, відтворення аварійної ситуації з метою з'ясувати, як повинен вести себе автомобіль, щоб мінімізувати її наслідки для водія, пасажирів і пішоходів, тобто для всіх учасників інциденту [1]. Щоб перевірити безпеку автомобілів їх доводиться розбивати, моделюючи особливості різних видів зіткнень. Вимоги до безпеки автомобілів з кожним роком підвищуються, а значить, і ускладнюються краш-тести. Якщо в перших краш-тестах автомобілі випробовували тільки на фронтальний удар, то незабаром з'явилися і бічні, і хлистів (ззаду). Параметрів оцінки - більше сотні, а в майбутньому їх може стати значно більше. За результатами випробувань автомобілі отримують зірки і рейтинги і ми можемо судити наскільки безпечна нова або стара машина. Автомобілі з'явилися і завоювали популярність, а їх безпеку ніяк не оцінює. І значно пізніше, коли мільйони машин зайняли дороги різних країн і континентів, їх безпека не вивчалася і не порівнювалася. Кількість загиблих і травмованих в аваріях стало рости, як і кількість автомобілів на дорогах, і проблему безпеки автотранспорту стали вивчати з одного боку автовиробники, з іншого боку державні або добровільні об'єднання автомобілістів [2].

Існує кілька організацій які проводять краш - тести, і у кожного свої стандарти [3]:

- IIHS (США);
- NHTSA (США);
- ANCAP (Австралія);

- EuroNCAP (Євросоюз);
- R NCAP (Японія).

Самою впливовою в цьому списку є EuroNCAP (Євросоюз) За системою краш - тесту EuroNCAP (Євросоюз), в машину встановлюють манекени [1]-[2].

При пошкодженні манекена по системі EuroNCAP автомобілю знімають бали, чим більше зняли тим менше залишилося. Бали представлені в якості в зірках і оцінка відбувається з п'яти зірок, це максимальне значення. Найпоширенішим видом краш-тесту є лобове зіткнення, розігнаний автомобіль б'ють в бетонний блок. Зазвичай швидкість автомобіля дорівнює 100 км/ч, що еквівалентно 200 км/год при реальних аваріях, коли автомобіль б'ється в автомобіль, що стояв. За системою EuroNCAP, лобовий удар відбувається при швидкості в 60 км/ч.

Подібні дослідження потребують великих обчислювальних потужностей. Для виконання високопродуктивних розрахунків, а також обробки великих масивів даних, як правило, використовують суперкомп'ютери або обчислювальні кластери. Для збільшення продуктивності обчислювальні кластери об'єднують високошвидкісними каналами зв'язку в спеціалізовані грид-системи [3]-[4]. Тому завдання розроблення засобів для обробки та обчислення даних дослідження технічних об'єктів є актуальною.

1.2 Засіб обробки інформації - кластер

Кластер - група взаємно з'єднаних обчислювальних систем (вузлів), які працюють спільно і складають єдиний обчислювальний ресурс. Для зв'язку вузлів використовується одна зі стандартних мережевих технологій (Fast / Gigabit Ethernet, Myrinet) на базі шинної архітектури або комутатора. Приклади кластерних обчислювальних систем: NT-кластер в NCSA, Beowulf-кластери [4].

Спочатку перед кластерами ставилися два завдання: досягти великої обчислювальної потужності і забезпечити підвищену надійність системи.

Архітектура кластерних систем багато в чому схожа на архітектуру MPP-систем [5]. Той же принцип розподіленої пам'яті, використання в якості обчислювальних вузлів закінчених обчислювальних машин, великий потенціал для масштабування системи і цілий ряд інших особливостей. У першому наближенні кластерну технологію можна розглядати як розвиток ідей масових паралельних обчислень. З іншого боку, багато рис кластерної архітектури дають підставу вважати її самостійним напрямком в області MIMD-систем.

Як вузла кластера може виступати як однопроцесорний комп'ютер, так і системи типу SMP (SMP-система представляється як єдина машина). Як правило, це не спеціалізовані пристрої, пристосовані під використання в обчислювальній системі, як в MPP, а серійно випускаються обчислювальні машини і системи. Ще одна особливість кластерної архітектури полягає в тому, що в єдину систему об'єднуються вузли різного типу, від персональних комп'ютерів до потужних обчислювальних систем. Кластерні системи з однаковими вузлами називають гомогенними кластерами, а з різнотипними вузлами - гетерогенними кластерами [3]-[5].

Використання машин масового виробництва істотно знижує вартість обчислювальних систем, а можливість варіювання різних за типом вузлів дозволяє отримати необхідну продуктивність за прийнятну ціну. Важливо і те, що вузли можуть функціонувати самостійно і окремо від кластера. Для цього кожен вузол працює під управлінням своєї операційної системи. Найчастіше використовуються стандартні ОС: Linux, FreeBSD, Solaris і версії Windows [4].

Вузли в кластерній системі об'єднані високошвидкісною мережею. Рішення можуть бути простими, заснованими на апаратурі Ethernet, або складними з високошвидкісними мережами пропускної здатності в сотні мегабайтів в секунду (Мбіт/с). До останньої категорії відносяться мережі SCI

компанії Scali Computer (100 Мбіт/с) і Mirynet (120 Мбіт/с). В принципі, за основу кластерної системи може бути взята стандартна локальна мережа (або мережу більшого масштабу), зі збереженням прийнятих в ній протоколів (правил взаємодії). Апаратурні зміни можуть бути зайвими або, в гіршому випадку, зводяться до заміни комунікаційного обладнання на більш продуктивне. При з'єднанні машин в кластер майже завжди підтримуються прямі межмашинного зв'язку [6].

Обчислювальні машини (системи) в кластері взаємодіють відповідно до одним з двох транспортних протоколів. Перший з них, протокол TCP (Transmission Control Protocol), оперує потоками байтів, гарантуючи надійність доставки повідомлення. Другий - UDP (User Datagram Protocol) намагається посилати пакети даних без гарантії їх доставки. Останнім часом застосовують спеціальні протоколи, які працюють набагато краще, наприклад Virtual Interface Architecture (VIA) [7].

При обміні інформацією використовуються два програмних методу: передачі повідомлень і розподіленої, спільно використовуваної пам'яті. Перший спирається на явну передачу інформаційних повідомлень між вузлами кластера. В альтернативному варіанті також відбувається пересилання повідомлень, але рух даних між вузлами кластера приховано від програміста.

Підключення вузлів до мережі здійснюється за допомогою мережевих адаптерів. З огляду на роль комунікацій, для зв'язку ядра обчислювального вузла з мережею використовується виділена шина введення / виводу. До цієї шині також підключаються локальні магнітні диски. Наявність таких дисків типова для кластерних систем, але не характерно для MPP-систем. Елементи обчислювального ядра об'єднуються за допомогою локальної системної шини. Зв'язок між цією шиною і шиною введення / виведення забезпечує міст. Схема кластера представлено на рисунку 1.1.

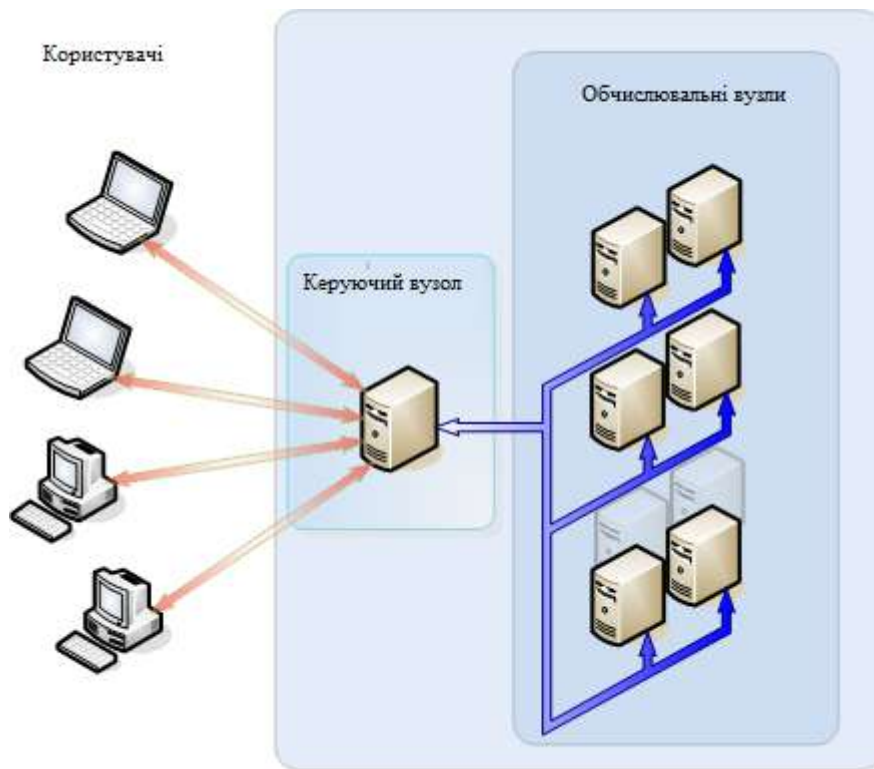


Рисунок 1.1 – Схема кластера

Кластери забезпечують високий рівень доступності - в них відсутні єдина операційна система і спільно використовувана пам'ять, тобто немає проблеми когерентності кешей [5]-[7].

При створенні кластерних систем використовується один з двох підходів.

Перший підхід застосовується для побудови невеликих кластерних систем, наприклад, на базі невеликих локальних мереж організацій або їх підрозділів. В такому кластері кожна машина продовжує працювати як самостійна одиниця, одночасно виконуючи функції вузла кластерної системи [4].

Другий підхід орієнтований на використання кластерної системи в ролі потужного обчислювального ресурсу. Вузлами кластера служать тільки системні блоки обчислювальних машин, компактно розміщуються в спеціальних стійці. Управління системою і запуск завдань здійснює повнофункціональний хост-комп'ютер. Він же підтримує дискову підсистему

кластера і різноманітне периферійне устаткування. Відсутність у вузлів власної периферії істотно здешевлює систему [3].

Чотири переваги, що досягаються за допомогою кластерів [3]-[6]:

- абсолютна масштабованість. Можливе створення великих кластерів, що перевершують по обчислювальній потужності навіть найпродуктивніші поодинокі обчислювальні машини. Кластер в змозі утримувати десятки вузлів, кожен з яких представляє собою мультипроцесор;

- нарощувана масштабованість. Кластер будується так, що його можна нарощувати, додаючи нові вузли невеликими порціями. Таким чином, користувач може почати з помірною системою, розширюючи її в міру необхідності;

- високий коефіцієнт готовності. Оскільки кожен вузол кластера – самостійна машина або система, відмова одного з вузлів не призводить до втрати працездатності кластера. У багатьох системах відмовостійкість автоматично підтримується програмним забезпеченням;

- співвідношення ціна / продуктивність. Кластер будь-якої продуктивності можна створити, поєднуючи стандартні «блоки», при цьому його вартість буде нижче, ніж у одиночному комп'ютері з еквівалентною обчислювальною потужністю.

У той же час основний недолік, властивий кластерним системам, - взаємодія між вузлами кластера займає набагато більше часу, ніж в інших типах обчислювальних систем.

Але існують системи в яких можна об'єднати робочі станції, які використовуються для різних завдань.

1.3 Засіб обробки інформації - суперкомп'ютер

Супер комп'ютер - обчислювальна машина, значно перевершує за своїми технічними параметрами більшість існуючих

комп'ютерів. Компанія NCAS в своїх розрахунках використовувала суперкомп'ютер з такими технічними характеристиками [8]:

- побудований на двох дванадцяти ядерних процесорах - Intel Xeon E5-2697V2 (ядро Ivy Bridge-EP). Разом 24 реальних ядра і 48 віртуальних ядер. Кожне ядро працює на частоті - 2.7 ГГц;
- в основі суперкомп'ютера - материнська плата ASUS Z9PE-D16 на чіпсеті Intel® C602 chipset. Серверна плата Z9PE-D16 є сумісною з процесорами серії Intel Xeon E5-2600 першого і другого поколінь (Ivy Bridge-EP), пропонує 16 слотів DIMM для установки до 512 гігабайт оперативної пам'яті і підтримує шину PCI Express 3.0;
- найвищу продуктивність в обчислювальних задачах забезпечують: обчислювальний процесор NVIDIA TESLA K40 в тандемі з професійною відеокартою Quadro K6000;
- 256Gb оперативної пам'яті DDR3 Kingston діючу пенсійну систему чотирьох канальному режимі. Ми використовуємо тільки спеціальну пам'ять ECC - з автоматичною корекцією помилок;
- дискова підсистема: два швидкодіючих накопичувача SSD, об'ємом 480Gb кожен і два продуктивних жорсткі диски по 4Tb з інтерфейсом SATA3 6Gb;
- вартість 535 тис грн.

Таким чином, виходячи з технічних характеристик та фінансової змоги для подальшої роботи було прийнято рішення працювати на базі кластеру.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Існуючі аналоги системи

Дана система призначена для моделювання та розрахунків безпеки автомобіля. Данні розрахунки потребують великих обчислень. в Інституті кібернетики ім. В.М. Глушкова НАН України створено і введено в дослідну експлуатацію високоефективні обчислювальні системи - суперкомп'ютери з кластерної архітектурою SKIT-1, SKIT-2, SKIT-3, SKIT-4 на базі сучасних мікропроцесорів фірми Intel [9].

2.1.1 Суперкомп'ютерний обчислювальний центр SKIT – 1

SKIT – 1 [9] - 24-вузловий кластер на мікропроцесорах Intel Xeon

Пікова продуктивність кластера 255 Гфлопс (1 Гфлопс = 1 млрд. операцій з плаваючою комою в секунду), реальна продуктивність 189 Гфлопс.

Тип процесора - однопоточний 32-розрядний Intel Xeon (тактова частота - 2,67 ГГц, кеш - 512 Кбайт, споживана потужність 60-100 Вт, число процесорів у вузлі кластера - 2).

Число ядер -процесорів у вузлі - 2. Оперативна пам'ять вузла - 2 Гбайт. Система зберігання даних - обслуговує всі кластери, типу RAID5, глобальна файлова система загальним обсягом 20 Тбайт.

Загальна споживана потужність - 8 кВА від мережі 380 В.

Обчислювальні засоби кластера (обчислювальні вузли і керуючий вузол) являють собою масив серверів, зв'язаних між собою двома ЛВС - високошвидкісною мережею Infiniband (пропускна здатність каналу 800 Мбайт/с) і мережею Gigabit Ethernet (пропускна здатність каналу до 1000 Мбіт/с).

Мережа Infiniband призначена для високошвидкісного обміну між вузлами в ході обчислень. При обміні даними між двома вузлами по мережі

Infiniband з використанням протоколів MPI може бути досягнута пропускна здатність 750 Мбайт/с.

2.2.2 Суперкомп'ютерний обчислювальний центр СКІТ – 2

СКІТ – 2 [9] - 32-вузловий кластер на мікропроцесорах Intel Itanium2/ Пікова продуктивність кластера 360 Гфлопс, реальна продуктивність 280 Гфлопс. Сеть Gigabit Ethernet призначена для з'єднання всіх обчислювальних вузлів кластера з керуючим вузлом і файл-сервером.

Тип процесора - одноядерний 64-розрядний Intel Itanium2 (тактова частота - 1,4 ГГц, кеш - 3 Мбайт, споживана потужність 135 Вт, число процесорів у вузлі кластера - 2).

Оперативна пам'ять вузла - 2 Гбайт. Число ядер-процесорів у вузлі – 2. Система зберігання даних - типу RAID5, глобальна файлова система загальним обсягом 20 Тбайт.

Загальна споживана потужність - 20 кВА від мережі 380 В.

Обчислювальні засоби кластера (обчислювальні вузли і керуючий вузол) являють собою масив серверів, зв'язаних між собою двома ЛВС - високошвидкісною мережею SCI (пропускна здатність каналу 350 Мбайт/с) і мережею Gigabit Ethernet (пропускна здатність каналу до 1000 Мбіт/с).

Мережа SCI призначена для високошвидкісного обміну між вузлами в ході обчислень. При обміні даними між двома вузлами по мережі SCI з використанням протоколів MPI може бути досягнута пропускна здатність 345 Мбайт/с.

2.2.3 Суперкомп'ютерний обчислювальний центр СКІТ – 3

СКІТ – 3 [9] - 127-вузловий кластер на багатоядерних процесорах (75 вузлів на двоядерних процесорах Intel Xeon 5160 і 52 вузла на чотирьохядерних процесорах Xeon 5345).

Пікова продуктивність кластера 7500 Гфлопс, реальна підтверджена продуктивність (на 125 вузлах) 5317 Гфлопс.

Тип процесорів - двоядерний Intel Xeon 5160 і чотирьохядерний Intel Xeon 5345 з архітектурою EM64T (тактова частота відповідно - 3,0 ГГц і 2,2 ГГц, кеш - 4 Мбайт, споживана потужність 80 Вт, число процесорів у вузлі кластера - 2).

Оперативна пам'ять вузла - 2 Гбайт на ядро, відповідно, 8 і 16 Гбайт.

Число ядер-процесорів у вузлі відповідно - 4 і 8, всього в кластері 716 ядер.

Система зберігання даних - типу RAID5, глобальна файлова система Lustre загальним обсягом (у дзеркалі) 20 Тбайт.

Загальна споживана потужність - 60 кВА від мережі 380 В.

Обчислювальні засоби кластера (обчислювальні вузли і керуючий вузол) являють собою масив серверів, зв'язаних між собою двома ЛВС - високошвидкісною мережею InfiniBand (пропускна здатність каналу до 900 Мбайт/с) і мережею Gigabit Ethernet (пропускна здатність каналу до 1000 Мбіт/с).

Мережа InfiniBand призначена для високошвидкісного обміну між вузлами в ході обчислень. При обміні даними між двома вузлами по мережі InfiniBand з використанням протоколів MPI може бути досягнута пропускна здатність 850 Мбайт/с.

Мережа Gigabit Ethernet призначена для з'єднання всіх обчислювальних вузлів кластера з керуючим вузлом і файл-сервером.

2.2.4 Суперкомп'ютерний обчислювальний центр СКІТ – 4

СКІТ – 4 [9] - 28-вузловий кластер на багатоядерних процесорах Intel Xeon E5-2600

Кластер базується на новітній платформі HP ProLiant Gen8 BladeSystems і має наступні характеристики:

- кластер складається з 28 вузлів на базі центральних процесорів Intel Xeon E5-2600 з частотою 2.6 ГГц, має 448 обчислювальних ядер, 36 прискорювачів NVidia Tesla M2075, 1,8 ТБ оперативної пам'яті;
- інтегрований із загальним сховищем даних кластерного комплексу обсягом 120 ТБ;
- мережу передачі даних між вузлами Infiniband FDR 56 Гбіт/с.

Суперкомп'ютер в своєму складі має як класичні вузли з центральними процесорами, так і гібридні вузли з графічними прискорювачами.

Кожен вузол має 16 ядер (32 в режимі HyperThreading), 64 ГБ оперативної пам'яті. Гібридні вузли додатково мають 3 прискорювача NVidia Tesla M2075.

Загалом кластер має такі характеристики продуктивності:

- пікова продуктивність 30 ТФлопс;
- реальна продуктивність 18 ТФлопс.

СКІТ-4 інтегрований з високопродуктивним сховищем даних обсягом 120 ТБ на основі паралельної файлової системи Lustre.

Основним недоліком даних систем є платний контент та залежність користувача від сторонньої апаратури.

2.2 Вибір апаратної та програмної частини для дослідження

2.2.1 Порівняння процесорів

При поліпшенні продуктивності кластерних систем виникає необхідність в порівнянні процесорів. Для тестування було обрано два типи доступних для тестування процесорів: Core I5 а також P4.

Для тесту була обрана задача, яка була сформована в LS-DYNA з такими вихідними даними: 269,249 елементів, пов'язаних 285,832 вузлами [10]. Симуляція аварії проводилась на швидкості 56 км \ год, час виміру до аварії 70мс, тривалість всього виміру 150мс.

На першому етапі вимірів було виявлено, що використовуючи мінімальний кластер, що складається з двох нодів виходить результат, який показує відмінність обчислювальної потужності двох різних типів процесорів. У першому випадку процесори P4 показали результат часу обчислення, який склав 23,2 тис. секунд, тим часом кластер складається з процесорів Core I5 покоління перевершив P4 впоравшись із завданням за 18,3 тис. секунд.

Далі були отримані співвідношення 11,8 тис. і 10,3 тис. Для 4-х процесорних кластерів, 6,2 тис. і 5,3 тис. для 8-ми процесорних систем, 4 тис. і 3,75 тис. для 16 нодів кластера і в завершенні 2,3 тис. і 2,1 тис. секунд рахунки для 32 процесорної системи. З розрахунків можна зробити висновок, що з кожним подвійним збільшенням кількості процесорів, виходить лінійне зменшення відносини різниць між двома групами процесорів, що відрізняються заявленою частотою і потужністю. В результаті проведених тестів і порівнянь було прийнято рішення зупинити свій вибір на процесорах CoreI5.

Все забезпечення мережі поділяють на два види: апаратне і програмне. Щоб мережа функціонувала потрібні сервера, комп'ютери абонентів, пристрої для об'єднання комп'ютерів в мережі і лінії зв'язку між ними.

Для вибору ПК, було запропоновано рішення у вигляді комп'ютерного класу, в якості робочих станцій були обрані персональні комп'ютери з такими технічними характеристиками:

- процесор Core i5 10400F. кількість потоків 12;
- кількість ядер 6 ядер;
- тактова частота ядра 2.9 ГГц в режимі Turbo 4.3 ГГц;
- тип роз'єму Socket 1200;
- об'єм кеш-пам'яті 2-го рівня 1.5 МБ;
- об'єм кеш-пам'яті 3-го рівня 12 МБ;
- частота шини даних 8 ГТ / с;

- материнська плата: H410 mATX;
- оперативна пам'ять: 16 Гб;
- тип пам'яті DDR4-2666 MHz;
- відеокарта GeForce RTX 2060 6 Gb;
- обсяг відеопам'яті 6144 Mb;
- жорсткий диск HDD 1000 Гб SATA III.

2.2.2 Порівняння і тестування технологій

Навіть при рівномірному розподілі потужності може виникнути втрата в швидкості рахунку. Якщо у кластера є велика розносна сітка обчислень йому доведеться витратити більше часу на пересилку даних між нодами і доменом ніж на час обрахунку однієї ітерації, в цьому випадку на швидкість рахунки вплине безпосередньо локальна мережа.

Найдоступніші технології, які можна використовувати - Ethernet, Fast Ethernet, Gigabit Ethernet. Для тестування була використана завдання, яка була сформована в LS-DYNA з такими вихідними даними: 269,249 елементів, пов'язаних 285,832 вузлами. Симуляція аварії проводилась на швидкості 56 км \ год, час виміру до аварії 70мс, тривалість всього виміру 150мс.

Тестування було розпочато на 2-х процесорної системі і був показаний однаковий результат для всіх типів мереж, який склав 23,2 тис. секунд. Далі було отримано підтвердження того, що в даному випадку швидкість мережі має значення. І при випробуванні на 4-х процесорному кластері був отриманий результат 11,95 тис. для Ethernet, 11,8 тис. для Fast Ethernet і 11,8 для Gigabit Ethernet, на наступному етапі вже всі три типи мережі почали явно відрізнятися часом рахунки, різниця почала даватися ознаки в разі на 8-ми процесорному кластері. 6,35 тис. (E), 6,2 тис. (FA), 6,1 тис. (GE). Після чого був зроблений висновок, що пропускна здатність мережі може перешкоджати збільшенню продуктивності кластера не

дивлячись на частоту процесора і версію програми. Наступним кроком стало тестування продуктивності для 16-ти процесорної системи. 4,25 тис. для Ethernet мережі, 4 тис. рівно для Fast Ethernet і 3,85 для Gigabit Ethernet. На передостанньому етапі тестування видно явну перевагу між стандартами мережі Ethernet і Gigabit Ethernet. І нарешті був проведений тест, який показав максимальний розрив часу між стандартами, а саме 2,65 тис. для (E), 2,3 тис. для (FE) і 2,05к тис. для (GE). З цих 29% приросту швидкості між першими і останніми даними можна зробити висновок, що коли починає рости кількість нодів, або кількість даних пересилаються по мережі, то чим ширше буде пропускна здатність однієї, тим швидшим буде наш результат.

Після проведених тестів було прийнято рішення для підвищення продуктивності кластера вибрати технологію Gigabit Ethernet.

Провівши ряд тестів і знайшовши шляху поліпшення продуктивності кластера, і використовуючи сумарні знання про більш швидкі процесори, більш швидкі канали передачі даних, інтерфейс передачі повідомлень LAM / MPI, безпосередньо шини комп'ютерів, безпосередньо операційні системи (позначається найкраща оптимізація коду під Linux), пропускна здатність пам'яті, безпосередньо впливають на швидкість кластера. А для створення обчислювального кластера можна застосовувати такі інструменти, наприклад Conda і pip, щоб налаштувати середовище Python. Даний підхід дає можливість використовувати для кластеру як доступні машини так і Azure.

2.2.3 Програмний засіб для моделювання випробувань

Відомі програмами для кінцево-елементного аналізу - LS-DYNA і ABAQUS [10]-[11].

ABAQUS - це універсальна програма загального призначення для проведення багатоцільового інженерного аналізу міцності.

ABAQUS дозволяє розрахувати складні нелінійні напружено-деформовані стани будівельних споруд і оцінювати їх міцність і стійкість з урахуванням багатофакторного навантаження, в тому числі і сейсмічного, теплового та вибухового.

Для розрахунків в дипломному проєкті була використана програма LS-DYNA-TRIAL, яка доступна для академічних та освітніх установ та для студентів.. Дана програма призначена для вирішення тривимірних динамічних нелінійних задач механіки деформованого твердого тіла, механіки рідини і газу, тепло переносу, а також пов'язаних задач. LS-DYNA знайшла широке застосування в таких галузях науки і техніки, як автомобілебудування (симуляція краш-тестів), військово-промисловий комплекс (симуляція вибухів боєприпасів і їх вплив на навколишні предмети), авіа- і ракетобудування (проектування реактивних двигунів і сопел) [11].

У LS-DYNA реалізовані явний і неявний метод кінцевих елементів з можливістю побудови лагранжевої, ейлерової і гібридної сітки, багатокомпонентна гідродинаміка, безсітковий метод згладжених частинок, безсітковий метод, заснований на методі Гальоркіна. Програма має вбудовані процедури автоматичної перебудови і згладжування кінцево-елементної сітки при виродження елементів, високоефективні алгоритми розв'язання контактних задач, широкий набір моделей матеріалів, можливості для користувача програмування [10].

Доступно дві основні версії LS-DYNA. Для визначення більш правильного видання було проведено порівняння між двома версіями програми LS-DYNA 960SMP і LS-DYNA 960MPP (кластерна версія програми). Тести проводилися під операційною системою Linux (рекомендоване програмне забезпечення від виробника програми).

Для початку була обрана тестова модель Plymouth Neon (рис. 2.1), одна з небагатьох публічно доступних моделей для аналізу транспортних засобів, створена національним центром аналізу Краштест (NCAC) спільно з

університетом Джорджа Вашингтона, яка містить 269,249 елементів, пов'язаних 285,832 вузлами.



Рисунок 2.1 – Модель автомобіля

Даний автомобіль пройшов ряд точних вимірів і був перенесений у вигляді 3d зображення в програму (рис. 2.2).

Симуляція аварії проводилась на швидкості 56 км \ год, час виміру до аварії 70мс, тривалість всього виміру 150мс. [6]-[10]. Низче представлена розроблена структурна модель автомобіля на рис. 2.3 яка використовувалась для симуляції аварії, низче представлена модель до (рис. 2.4) та після удару (рис. 2.5).



Рисунок 2.2 – 3D модель автомобіля

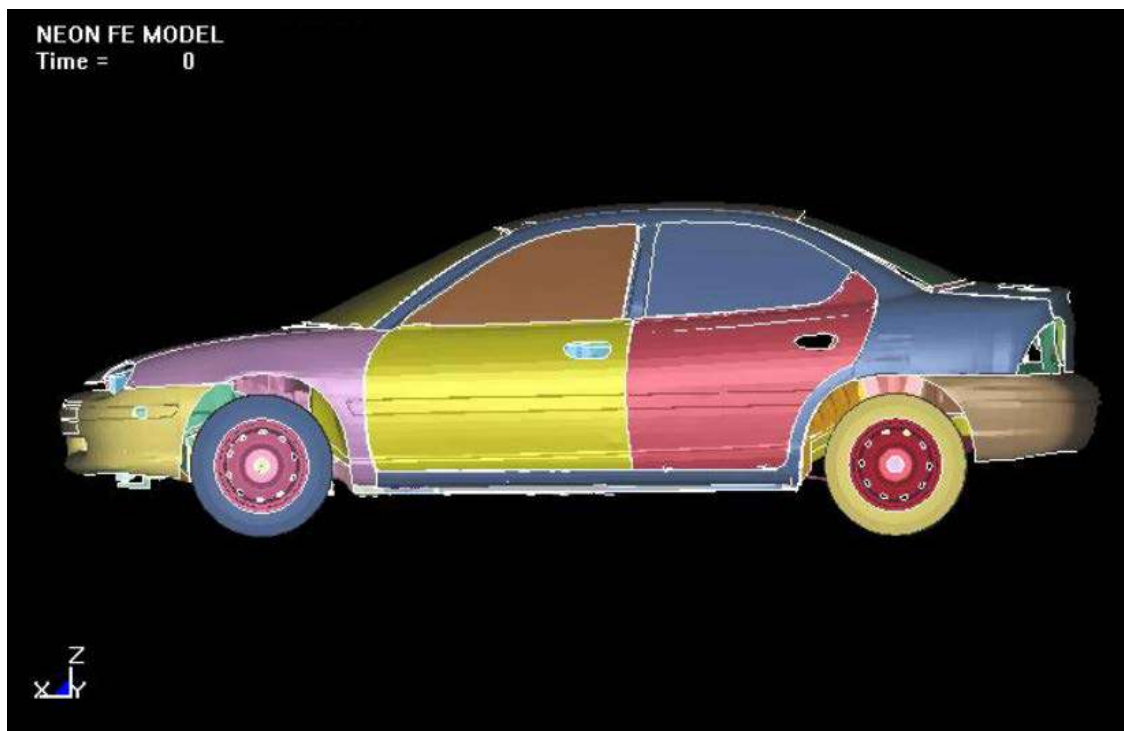


Рисунок 2.3 – Структурна модель

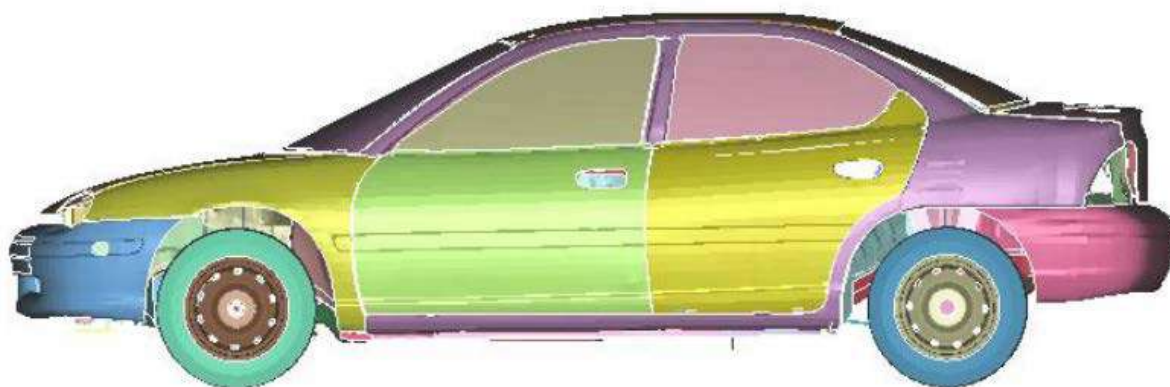


Рисунок 2.4 – Модель до удару

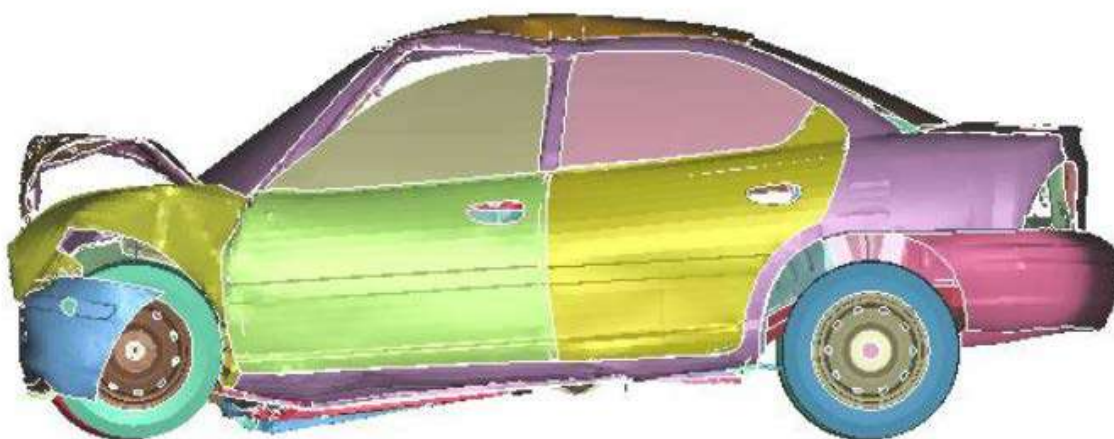


Рисунок 2.5 – Модель після удару

Для початку розрахунок був зроблений на одному процесорі, його результат склав 25,5 тис. секунд для SMP версії і 31,7 тис. для MPP версії. Розрахувавши співвідношення помітно що на даному етапі тестування SMP версія перевершує MPP на 24,3% що досить багато як для невеликої завдання. Далі розрахунок був зроблений на 2-х процесорному кластері і його результат склав 16,3 тис. секунд для SMP версії і 17,4 тис. секунд для MPP версії відповідно, що склало 6,7% приросту продуктивності SMP версії. Як можна передбачити майбутні результати повинні будуть зменшити, це співвідношення і навіть збільшити його, але вже в інший бік. Приступивши

до вимірів 4-х процесорного кластера отримано результат, який підтвердив припущення. SMP версія впоралася з наданою завданням за 8 тис. секунд а її «кластерна» версія за 7,8 тис. секунд. І так навіть на даному етапі видно маленький (2,5%) профіт MPP версії програми вже на 4 процесорах. Якщо справа вся справа тільки в кількості, то можна спробувати порівняти більш широкий спектр. Далі був заміряний 8-ми процесорна система яка дала результати - 5,05 тис. для SMP і 4,7 тис. для MPP версій. З ростом продуктивності кластера видно і його перевага для мережових версій програм. На даним момент приріст швидкості обрахунку інформації склав 7,4%, якби мова йшла про більш об'ємною сітці обчислень яка б затягувалася на годинник, то навіть цього показника вистачило б для обґрунтування того, що SMP версія більш придатна таких обчислень. Максимум за кількістю процесорів використовуваних в SMP версії даної програми - 16. Отже, після тестів результат SMP був 3,75 тис., а MPP - 2,2 тис. (70%). Далі був проведений тест, який показав подальшу можливість продуктивності MPP версії. SMP версія була запущена на 32 процесорах однак вона не показала збільшення продуктивності по відношенню до збільшеного кількості процесорів і залишилася на колишньому показнику, як і на 163,75 тис.. А MPP версія показала максимальну отриману в ході тестів продуктивність, яка виразилася в 1,42 тис. секунд. На прикладі даного тесту було отримано уявлення про можливості двох версій програми LS-DYNA, і за результатами тестування була обрана найбільш продуктивна версія MPP [11].

2.2.4 Вибір відкритої інфраструктури для розподілених обчислень

У даних тестах проводився розрахунок з обчислювальної сіткою 1,480,422 на 1,450,225 значень, наступні обчислення таких моделей як Toyota Camry будуть використовувати для обчислень сітку приблизно 1,900,000 на 1,900,000 значень.

Якщо розрахунки на кластері будуть проводитися більше 15 годин, а це максимальний період простою комп'ютерів поза робочим часом. Було прийнято рішення знайти програмне забезпечення, яке могло б проводити розрахунки, не заважаючи робочому процесу.

Найбільш відомою відкритою інфраструктурою для розподілених обчислень в даний момент вважається – BOINC [12].

В основі архітектури BOINC ідея кінцевого автомата - сервер складається з набору окремих підсистем, кожна з яких відповідає за свою цілком певну задачу, наприклад, виконання обчислень, передачу файлів і т.д. Кожна з підсистем перевіряє стан підзадачі, виробляє якісь дії і змінює стан підзадачі - так вони працюють в нескінченному циклі.

Система складається з сервера BOINC (при необхідності, розподіленого на кілька фізичних серверів з метою підвищення продуктивності, відмовостійкості та безпеки), безлічі клієнтів, які виконують завдання сервера і, можливо, додаткових компонент у вигляді приєднаних GRID-мереж.

Сервер BOINC складається з [13]:

- Web-сервера, який обробляє вхідні та вихідні повідомлення;
- сервера баз даних, який відстежує стан подзадач і відповідні їм результати, а також зберігає інформацію про клієнтів і набраних ними балах;
- п'яти різних демонів, які періодично перевіряють стан бази даних і виконують необхідні роботи з обслуговування системи і розподілу підзадач.

Основними компонентами BOINC сервера є [12]-[13].

1. Web-сервер не є обов'язковою частиною інфраструктури сервера BOINC. Однак його наявність обумовлена тим, щоб залучити учасників до проєкту. Так як на базі Web-сервера можна створити сайт з допомогою якого можна розширити коло учасників. Кожен проєкт розподілених обчислень на базі BOINC надає своїм учасникам можливість об'єднатися в команди і стежити за збільшенням числа набраних балів. Web-сервер може бути

реалізований на іншому сервері, єдина умова він повинен бути пов'язаний з базою даних сервера BOINC.

2. База даних - це основа всього проєкту BOINC, в ній зберігається вся інформація, що відноситься до сервера BOINC, включаючи відомості про зареєстрованих користувачів і пов'язаних з ними хостами, про додатки і версіях додатків, про клієнтів BOINC і їх версіях, а також про підзадачі і результати обчислень. До цієї інформації звертаються (як для читання, так і для внесення змін) спеціальні службові програми сервера. Система BOINC спочатку орієнтована на роботу з СУБД MySQL.

3. Служба обробки стану підзадач (Transitioner) - служба призначена для обробки стану обчислювальних підзадач і результатів їх вирішення. Вона не залежить від програми і є однаковою для всіх проєктів. Служба обробки перевіряє поточний стан підзадачі в базі даних і оновлює відповідні поля, коли підзадача готова перейти в новий стан. Складність полягає в тому, що підзадачі мають не стану як такі, а набори «підстанів». Ці підстани включають в себе також стану відповідних результатів обчислень. Наприклад, якщо результати готові до перевірки і до цього моменту досить даних для організації перевірки кворумом, то стан підзадачі змінюється на «готова до перевірки». Служба обробки є однією з найбільш ресурсномістких з точки зору навантаження на процесор, тому вона може бути розділена на кілька процесів, кожен з яких відповідає за певний набір підзадач. Відповідно, ці процеси можуть працювати на різних фізичних серверах.

4. Служба перевірки результатів (Validator) - для організації перевірки отриманих результатів. З метою забезпечення достовірності кожна з підзадач розраховується на кількох різних клієнтів. Тому отримані в результаті результати необхідно перевірити, звіривши між собою і визначивши «канонічне» рішення - результат, отриманий кворумом клієнтів. При цьому алгоритм перевірки результатів цілком залежить від розв'язуваної задачі - в одних досить упевнитися в рівність двох чисел, а в інших необхідно поелементно порівняти з десятком матриць. За реалізацію такого алгоритму

перевірки результату і відповідає служба. Крім того, програма перевірки може стежити за правдоподібністю результатів. Наприклад, якщо моделюється падіння тіла під впливом сили тяжіння, можна перевірити, чи не є кінцева точка вище над землею, ніж початкова. Отже, такі результати є свідомо неправильними і не повинні прийматися до уваги. При запуску служба звертається до бази даних для отримання інформації про нові результати, які потребують перевірки. Якщо такі є, то служба перевірки запускає відповідну функцію порівняння результатів. Для кожної глобального завдання, розв'язуваної за допомогою розподілених обчислень на базі BOINC, необхідно розробити дві функції в рамках служби перевірки: одна функція порівнює два результату, а інша - набори результатів. Перша використовується для прийняття рішення про надання очок, коли клієнтським додатком переданий новий результат і знайдено канонічне рішення. Друга функція використовується для визначення самого канонічного результату з набору результатів, переданих декількома клієнтами. Кількість результатів, необхідних для вироблення канонічного рішення, визначається при створенні підзадачі. Це значення може бути встановлено для всього програми в цілому, але також можливо вказати різні значення для різних клієнтів.

5. Служба освоєння (Assimilator) - програма освоєння періодично перевіряє наявність завершених підзавдань. Адміністратор проєкту повинен створити функцію, яка визначає, що необхідно зробити з канонічними результатами. Наприклад, їх можна архівувати і відсилати по електронній пошті певній людині або автоматично запускати подальшу обробку даних, виділяючи цікавлять фрагменти і записуючи їх на пристрій зберігання. Підзадача позначається як завершена тільки після проходження через службу освоєння.

6. Служба видалення файлів (File deleter) - це «збирач сміття» проєкту BOINC, вона просто перевіряє наявність завершених і освоєних підзадач, а при знаходженні таких очищає вхідні і вихідні файли сервера,

пов'язані з цими підзадачами. Отже, необхідно, щоб вихідні файли, що містять канонічний результат, десь зберігалися на фазу освоєння. При необхідності видаляти можна тільки файли, залишаючи записи в базі даних, тоді завжди можна буде пройти по базі даних і знайти інформацію про підзадача, результати і т.д. навіть після завершення підзадач (і видалення файлів).

7. Служба подачі (Feeder) і Планувальник (Scheduler) - вона завантажує необроблені підзадачі з бази даних в сегмент розділяється пам'яті. Ця попередня робота виконується для підвищення продуктивності системи BOINC в цілому за допомогою обмеження числа запитів до бази даних.

Планувальник - це CGI-програма, яка запускається, коли до сервера проєкту приєднується клієнт і запитує порцію завдань. Замість запиту до БД планувальник отримує завдання з сегмента розділяється пам'яті, в який завдання завантажує служба подачі. Планувальник має можливості за самостійним призначенням підзадач клієнтам, так як не всі клієнти мають однакові настройки і комп'ютери. Наприклад, один користувач може мати Linux-версію клієнта і виділити 100 Мб дискового простору і 200 Мб оперативної пам'яті, а інший клієнт може запустити тільки Windows-версію і дозволити використання не більше 10 Мб дискового простору і 10 Мб оперативної пам'яті. В цьому випадку планувальник приймає рішення про призначення більш об'ємного завдання Linux-клієнту, залишаючи найбільш прості підзадачі «слабкого» Windows-клієнта. Під час сесії роботи з планувальником клієнт також звітує про завершені роботи, які були вже завантажені на сервер з часу останньої сесії планування. В результаті клієнт залишається зі списком завдань на обробку та списком адрес, з яких можна отримати необхідні файли, тобто вхідні файли і файли програми, якщо їх немає на клієнтському комп'ютері.

8. Міст (Bridge) - служба, яка забезпечує зв'язок і спільну роботу над проєктом інфраструктури BOINC і стандартної GRID (наприклад, на базі популярної технології Globus Toolkit), заслуговує на окрему згадку.

Додатки BOINC спеціально розробляються під архітектуру BOINC. Вони викликають функції BOINC через інтерфейси, реалізовані в клієнті і виконують таку специфічну роботу як, наприклад, дозвіл імен файлів. Як наслідок, підзадачі проєкту BOINC не можуть бути безпосередньо (без додаткових модифікацій) запущені для розрахунку на інфраструктурі Grid. Такі проблеми взаємодії різних архітектур розподілених обчислень вимагають реалізації додаткових механізмів, які дозволяли зв'язку BOINC-Grid. Ці завдання і вирішує програмний міст, при цьому фактична реалізація моста може залежати від особливостей підключається Grid і проєкту, в рамках якого проводяться обчислення.

Основні етапи життєвого циклу завдання [11]:

- програма генерації завдань (розроблена спеціально для проєкту) створює підзадачі і необхідні для їх розрахунку вхідні файли;
- BOINC створює для кожного з підзавдань один або кілька однакових примірників (в залежності від налаштувань проєкту);
- планувальник BOINC розподіляє підзавдання різних клієнтських програм;
- кожна клієнтська програма завантажує з сервера вхідні файли;
- після розрахунку підзавдання клієнтська програма завантажує вихідні файли на сервер;
- клієнтська програма звітує про виконання підзавдання (можливо, після невеликої затримки, необхідної для зниження навантаження на програму-планувальник сервера);
- служба перевірки результатів (розроблена спеціально для проєкту) перевіряє вихідні файли, можливо, порівнюючи результати різних клієнтських програм;
- коли знайдено канонічне рішення, служба освоєння (розроблена спеціально для проєкту) обробляє результати, наприклад, поміщаючи їх в окрему базу даних або відсилаючи на електронну пошту;

– коли всі екземпляри підзавдання завершено, служба видалення файлів видаляє непотрібні більше вхідні і вихідні файли, а також очищає базу даних від інформації про підзавдання і його примірниках.

2.2.5 Мова програмування

Для формування кластеру та завдання необхідно розробити програмний додаток, тому необхідно обрати мову програмування.

Python є об'єктно-орієнтованою мовою програмування. Він надзвичайно простий і містить невелику кількість ключових слів, в той же час дуже гнучкий і виразний. Інтерпретатор Python реалізований практично на всіх платформах і операційних системах.

Зважаючи на те які програмна та апаратні засоби були обрані вибір був зупинений на Python. Дана мова програмування має всі необхідні бібліотеки та добре інтегрована з Voins.

Також необхідно провести зчитування файлів з LS-Dyna. Зважаючи на інтегрування мови до програмних додатків вибір був зупинений на C++.

3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

3.1 Запуск кластера

Після установки кластера, необхідно перевірити працездатність кластерної функції cluster-fork (рис.3.1). Вона дозволяє запускати один і той же додаток на всіх вузлах або підмножині вузлів кластера.

```
[root@test ~]# cluster-fork uname
compute-0-0:
Warning: Permanently added 'compute-0-0' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-1:
Warning: Permanently added 'compute-0-1' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-2:
Warning: Permanently added 'compute-0-2' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-3:
Warning: Permanently added 'compute-0-3' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-4:
Warning: Permanently added 'compute-0-4' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-5:
Warning: Permanently added 'compute-0-5' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-6:
Warning: Permanently added 'compute-0-6' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-7:
Warning: Permanently added 'compute-0-7' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-8:
Warning: Permanently added 'compute-0-8' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-9:
Warning: Permanently added 'compute-0-9' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-10:
Warning: Permanently added 'compute-0-10' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-11:
Warning: Permanently added 'compute-0-11' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-12:
Warning: Permanently added 'compute-0-12' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-13:
Warning: Permanently added 'compute-0-13' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-14:
Warning: Permanently added 'compute-0-14' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-15:
Warning: Permanently added 'compute-0-15' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-16:
Warning: Permanently added 'compute-0-16' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-17:
Warning: Permanently added 'compute-0-17' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-18:
Warning: Permanently added 'compute-0-18' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-19:
Warning: Permanently added 'compute-0-19' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-20:
Warning: Permanently added 'compute-0-20' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-21:
Warning: Permanently added 'compute-0-21' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-22:
Warning: Permanently added 'compute-0-22' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
compute-0-23:
Warning: Permanently added 'compute-0-23' (RSA) to the list of known hosts.
/usr/bin/ssh: creating new authority file /root/.Xauthority
linux
[root@test ~]#
```

Рисунок 3.1 – Запуск uname через cluster-fork

При першому виклику система додає вузли кластера в списки дозволених хостів. А при другому виконує безпосереднє завдання (рис 3.2).

```
[root@test ~]# cluster-fork uname
compute-0-0:
Linux
compute-0-1:
Linux
compute-0-2:
Linux
compute-0-3:
Linux
compute-0-4:
Linux
compute-0-5:
Linux
compute-0-6:
Linux
compute-0-7:
Linux
compute-0-8:
Linux
compute-0-9:
Linux
compute-0-10:
Linux
compute-0-11:
Linux
compute-0-12:
Linux
compute-0-13:
Linux
compute-0-14:
Linux
compute-0-15:
Linux
compute-0-16:
Linux
compute-0-17:
Linux
compute-0-18:
Linux
compute-0-19:
Linux
compute-0-20:
Linux
compute-0-21:
Linux
compute-0-22:
Linux
compute-0-23:
Linux
[root@test ~]#
```

Рисунок 3.2 – Другий запуск uname через cluster-fork

Після цього тесту видно що кластер запущений і працездатний. На наступному етапі було встановлено програмне середовище LS-DYNA.

3.2 Проєктування в програмному середовищі LS DYNA

Після збору кластера для визначення працездатності потужності і швидкості обчислювального вузла необхідно провести тестування. Для порівняння з обладнанням і продуктивністю була взята завдання, яке було обраховано National Crash Analysis Center (NCAC) , так як це найостанніша публічно доступна модель, яка підходить за складністю і обсягом обчислень [14]. Зібраний кластер будемо порівнювати з суперкомп'ютером від компанії SGI Altix 450 на базі процесорів Intel® Itanium 2, п'ятдесят шість 1,6 GHz \ 18Mb кеша процесорів створюють приблизно 206 Gflops обчислювальної

потужності, 112Gb ОЗУ і SAS диском розміром в 3,3 Тб з доступом на швидкості до 200Мб\с. Комп'ютер використовувався Університетом Джорджа Вашингтона, для обрахунку даного завдання, використовувалися тільки 24 процесора і комп'ютер впорався з нею за 2 години і 15 хвилин.

Основні дані:

- кількість деталей - 917;
- вузлів - 1,480,422;
- елементів - 1450,225;
- швидкість - 56, 3 км / год;
- 4324 променевих з'єднань;
- 423 вузлів твердого з'єднання;
- 16 вузлів рухомого з'єднання;
- 2862 точкових зварювання.

Далі на малюнках представлені різні види моделей для випробувань (рис.3.3 - 3.9).



Рисунок 3.3 – Задні сидіння



Рисунок 3.4 – Корпус машины

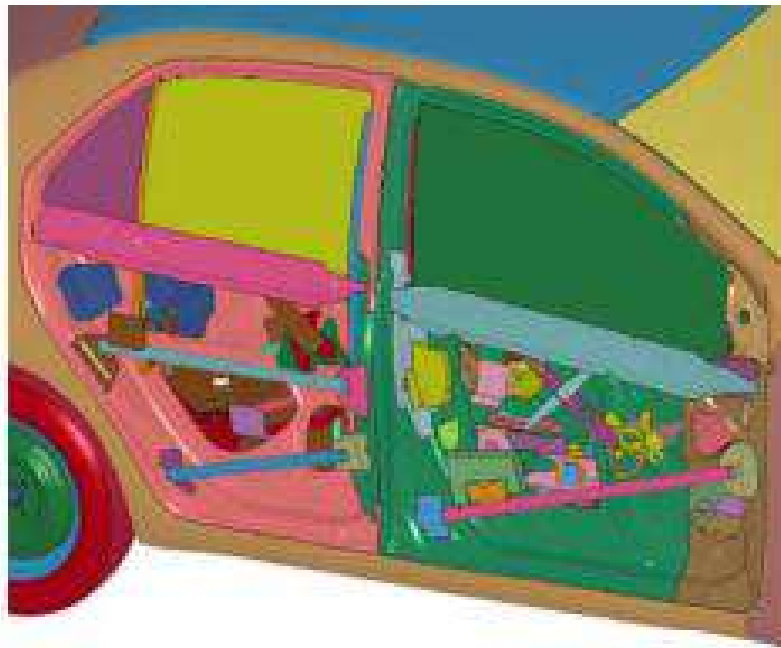


Рисунок 3.5 – Вид з боку



Рисунок 3.6 – Вид спереду



Рисунок 3.7 – Загальний вигляд

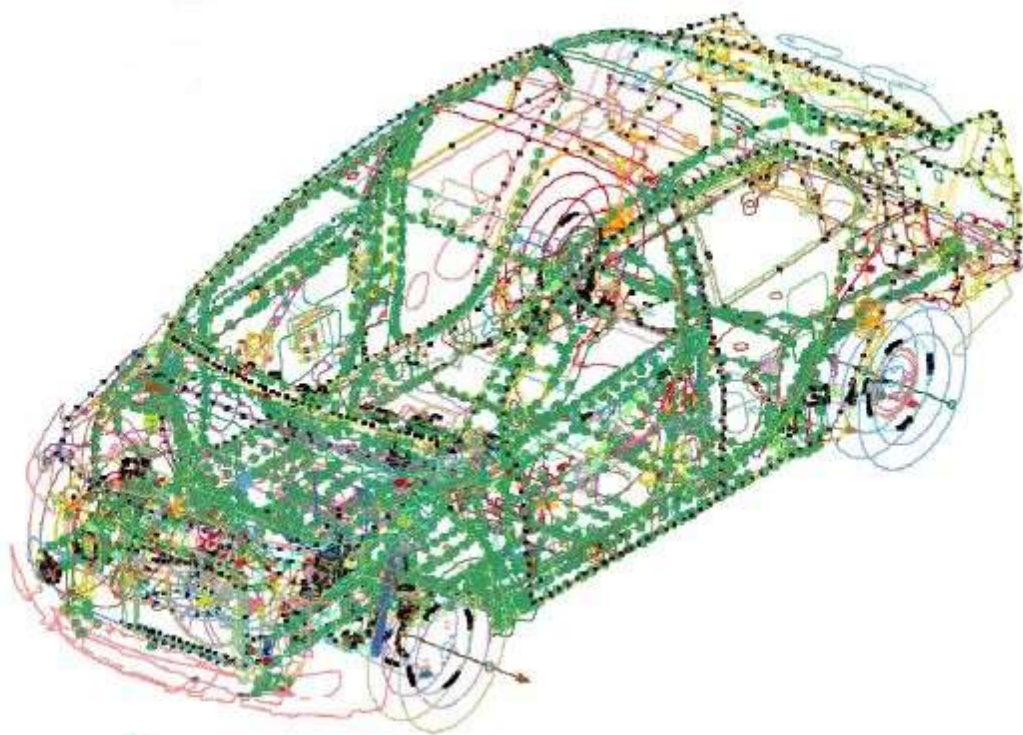


Рисунок 3.8 – Точковий вид

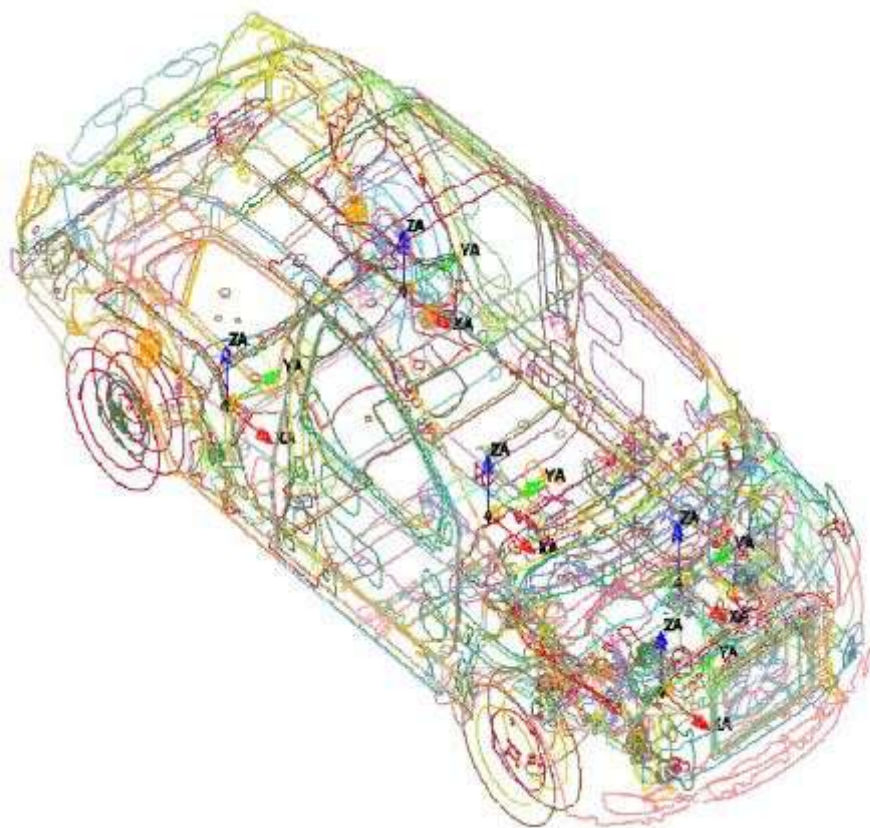


Рисунок 3.9 – Зображення вузлів, з'єднань

Акселерометри встановлені на лівому задньому сидінні, правому задньому сидінні вгорі і знизу двигуна.

У порівняннях беруть участь три моделі: комп'ютерна(рис. 3.14), седан (5677) і хечбек (6221). На малюнках зображені дані моделі до (рис. 3.10, рис. 3.12) і після удару (рис. 3.11, рис 3.13).



Рисунок 3.10 – Седан до удару



Рисунок 3.11 – Седан після удару



Рисунок 3.12 – Хечбек до удару



Рисунок 3.13 – Хечбек після удару

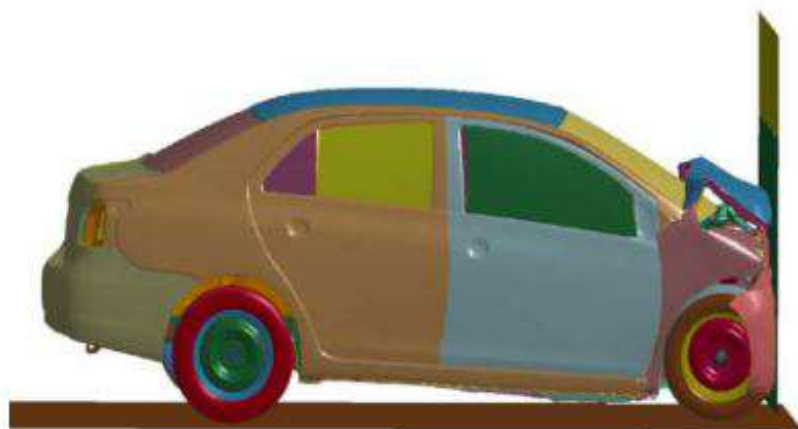


Рисунок 3.14 – Комп'ютерна модель після удару

Далі в таб. 3.1 наведені деякі дані про представлених вище моделях.

Таблиця 3.1 – Дані про моделі

	Комп'ютерна модель	Модель5677	Модель6221
Вага , кг	1263	1271	1245
Тип двигуна	1,5л	1,5л	1,5л
Розмір шин	60R15	60R15	60R15
Ширина колісної бази	2538	2551	2463
Тип	4-х дверний седан	4-х дверний седан	3-х дверний хечбек

Після проведення розрахунків було отримано безліч графіків, деякі з них представлені нижче на рис. 3.15:

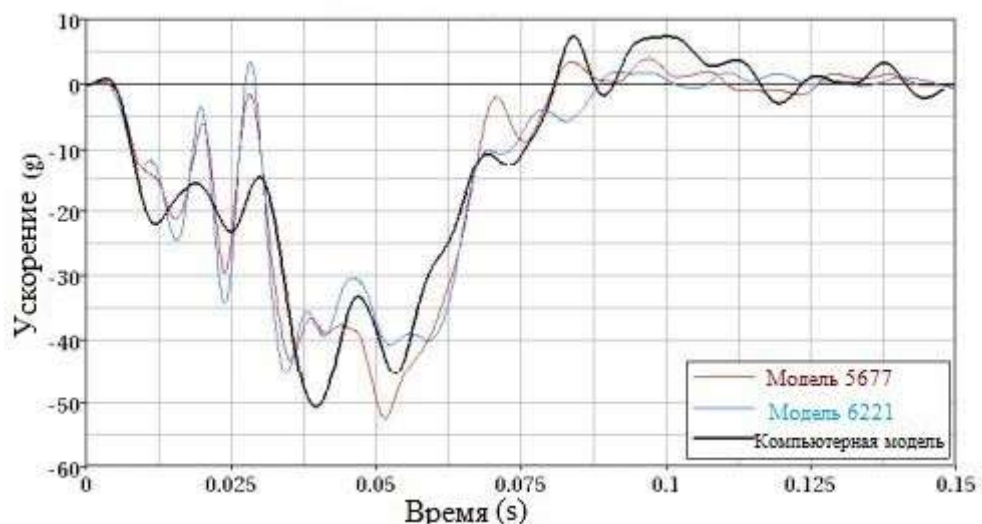


Рисунок 3.15 – Ліве заднє сидіння

На рис. 3.16 показана зміна прискорення при зіткненні в залежності від часу.

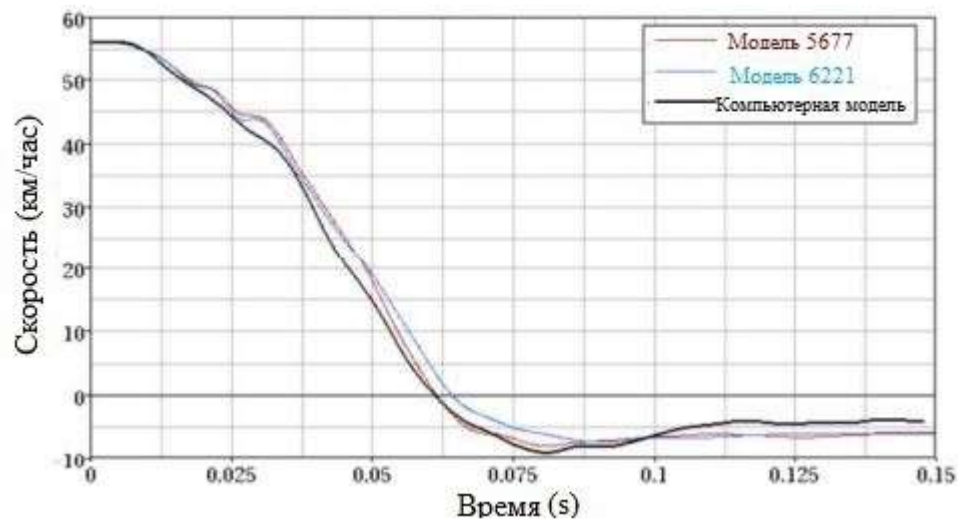


Рисунок 3.16 – Ліве заднє сидіння

Ідеальна обчислювальна потужність зібраного кластера становить 485 Гфлоп / сек, але в цьому кластері можливі втрати рівні 12% від ідеальної потужності. Таким чином, розрахунок на кластері завдання з кількістю вузлів 1,480,422 на 14 50,225 тривав 5 годині 54 хвилини, а на суперкомп'ютері 2 години 40 хвилин.

3.3 Зчитувач файлів

Програма містить зчитувач двійкових файлів результатів, що надходять з коду LS-DYNA. Основний код написаний на C ++ з використанням Python.

Результати повертаються до внутрішніх класів, таких як Node або Element, які можна отримати як за допомогою ідентифікатора, так і з частин. Для елементів оболонки, які мають кілька результатів (напруження та пластична деформація на шар інтеграції та 2х тензор деформації). Для імпорту даних використовуємо наступні бібліотеки:

```
#include "../dyna/dplot.h"

#include "../db/DB_Nodes.h"

#include "../db/DB_Elements.h"

#include "../db/Node.h"

#include "../db/Element.h"

#include <iostream>

#include <vector>

#include <set>
```

Для тестування класу Element використовуємо наступний фрагмент:

```
cout << endl << "> TESTS" << endl;

Element* element = d3plot->get_db_elements()-
>get_elementByID(SHELL,9770);

    cout << "Elem-ID:" << element->get_elementID() << endl;
    set<Node*> nodes = element->get_nodes();
    for(auto node : nodes) {
        cout << "Node-ID:" << node->get_nodeID() << endl;
        cout << "X:" << node->get_coords()[0] << endl;
        cout << "Y:" << node->get_coords()[1] << endl;
        cout << "Z:" << node->get_coords()[2] << endl;
    }
```

Результати повертаються до внутрішніх класів, таких як Node або Element, які можна отримати як за допомогою ідентифікатора. Повний зміст файлу тестування розміщений у додатку А.

3.4 Формування завдання в Voinc

На базі програмного комплексу Ansys, програмісти збирають модель транспортного засобу, вводячи дані по всім його точкам, створюють у мови руху, вектора, переш наслідком, і зберігають дані в . файл, який є вихідним для розрахунків. Наступною дією буде рішення математичних рівнянь стану твердих тіл , алго ритмів контактної взаємодії, рівняння нелінійної динаміки, руйнування, Ейлерової - Лагранджевої поведінки. Які будуть розраховуватися в межах на платформі Voinc і розраховані добровільними потужностями, потім вписані готовими вхідними даними в той же .k файл, з якого надалі програмний комплекс LS-Dyna проведе 3d векторизацію, вже з порахованими значеннями, що значно скоротить хід виконання останнього кроку . Таким чином провівши безпосередні розрахунки використовуючи технологію GRID буде значно полегшено завдання, що виконується на кластері, тим самим зменшується час проведення обчислень, однак збільшується час підготовки проєкту. Якщо розпланувати і скоординувати роботу конвеєрним чином, то кількість оброблюваної інформації можна значно підвищити, не дивлячись на збільшення часу розрахунків.

4 РОЗРОБКА ПРОГРАМИ

4.1 Налаштування кластеру

У каталозі верхнього рівня tutorial додайте новий файл Python з ім'ям 01-create-workspace.py, використовуючи наступний код. Задайте параметри (ім'я, ідентифікатор підписки, група ресурсів і розташування) відповідно до своїх вимог .

Лістинг 4.1

```
ws = _Workspace.create(name='<my_workspace_name>',
    subscription_id='<klafter-subscription-id>',
    resource_group='<myresourcegroup>',
    create_resource_group=True,
    location='<NAME_OF_REGION>')
ws.write_config(path='.klafter')
```

Файл .klafter / config.json містить метадані, необхідні для підключення до робочої області. А саме, він містить ідентифікатор підписки, групу ресурсів і ім'я робочої області.

Лістинг 4.2

```
ws = Workspace.from_config()
cpu_cluster_name = "cpu-cluster"
try:
    cpu_cluster = ComputeTarget(workspace=ws,
name=cpu_cluster_name)
    print('Found existing cluster, use it.')
except ComputeTargetException:
    compute_config =
AmlCompute.provisioning_configuration(vm_size='STANDARD
_D2_V2',
```

```

idle_seconds_before_scaledown=2400,

min_nodes=0,

max_nodes=24)

        cpu_cluster      =      ComputeTarget.create(ws,
cpu_cluster_name, compute_config)
        cpu_cluster.wait_for_completion(show_output=True)

```

4.2 Налаштування Boinc- сервера

Для установки і налагодження, може бути використаний практично будь-який комп'ютер в якості сервера BOINC, на якому буде формуватися основне завдання. Для роботи BOINC необхідно встановити допоміжне програмне забезпечення. Для розгортання сервера знадобляться СУБД MySQL, Web-сервер Apache, інтерпретатори і стандартні бібліотеки мов python і php.

Крім того, для отримання і складання вихідного коду будуть потрібні клієнт системи управління версіями Subversion (svn), компілятор gcc і засоби розробки зі складу GNU Tools, а також відкритий криптографічний пакет OpenSSL і допоміжна утиліта pkg-config.

Встановивши допоміжне програмне забезпечення, створюється користувач-адміністратор проєкту BOINC, (наприклад, boincadmin), від імені якого працюватимуть службові скрипти сервера. Цей користувач повинен мати можливість працювати з базою даних, а також створювати і видаляти файли в каталогах Web-сервера Apache.

Ще одна важлива вимога - сервер BOINC повинен мати доменне ім'я і «білий» IP-адреса.

Після того, як виконані всі допоміжні кроки і попередні умови, можна переходити безпосередньо до збірки програмних компонент сервера BOINC.

Для розгортання сервера BOINC, необхідно його скомпілювати. Необхідно завантажити вихідний код за допомогою команди `svn co` http://boinc.berkeley.edu/svn/branches/server_stable

Якщо все додаткові інструменти встановлені, то можна почати компіляцію (вихідний код вже повинен знаходитися в підкаталозі `server_stable` домашнього каталогу користувача `boincadm`).

```
boincadm@boincserver:~/server_stable> ./_autosetup
boincadm@boincserver:~/server_stable> ./configure
```

Компілюватиметься будуть серверні компоненти і додаткові бібліотеки - це буде написано в останньому рядку:

```
-- Build Components: ( libraries server) ---
```

Після компіляції слід запустити команду:

```
boincadm@boincserver:~/server_stable> make
```

Для виконання всієї необхідної роботи, пов'язаної зі створенням нового проєкту розподілених обчислень, розробниками BOINC створений спеціальний скрипт `make_project`. Знайти його можна в підкаталозі `tools` каталогу скомпільованої сервера.

Цей скрипт запускається з ім'ям нового проєкту:

```
boincadm@boincserver:~/server_stable> ./tools/make_p
project_toyota
```

Спочатку встановлюються параметри, пов'язані з доступом до Web-сторінці проєкту (тут `boincserver.domain.ru` - це ім'я сервера в мережі):

Creating project ' toyota ' (short name ' toyota '):

```

PROJECT_ROOT = /home/boincadm/projects/ toyota /
URL_BASE = http://boincserver.domain.ru/
HTML_USER_URL      =      http://boincserver.domain.ru/
toyota/
HTML_OPS_URL = http://boincserver.domain.ru/toyota
_ops/
CGI_URL      =      http://boincserver.domain.ru/toyota
_cgi/
KEY_DIR = /home/boincamd/projects/ toyota /keys/
DB_NAME = toyota
DB_HOST =

```

Потім створюються необхідні каталоги: Setting up server: creating directories

Генеруються криптографічні ключі: Keys don't exist in /home/boincadm/projects/ toyota /keys/; generate them? [Y/n] y

Setting up server files: generating keys

Далі копіюються необхідні файли, створюються таблиці бази даних і файли налаштувань, а також встановлюються служби, що не залежать від додатка:

Setting up server files: copying files

Setting up database

Setting up server files: writing config files

Setting up server files: linking cgi programs

update_translations finished

Done installing default daemons.

Скрипту `make_project` можливо вказати набір параметрів для зміни стандартних налаштувань. Найбільш затребуваними параметрами є `--delete_prev_inst` і `--drop_db_first`, що дозволяють переписати (або просто видалити) дані вже існуючого проєкту. Про решту можливостей можна дізнатися, використавши параметр `--help`.

У домашньому каталозі адміністратора проєкту BOINC є підкаталог `projects / toyota` наступного змісту: `boincadm@boincserver:~/server_stable> ls ~/projects/ Toyota`

Основним робочим каталогом бути каталог `/ home / boincadm / projects / toyota`, тому слід перейти в нього.

Установки Web-сервера Apache для нового проєкту вміщує в файлі `toyota.httpd.conf`. Його вміст можна або підключити в основному файлі налаштувань, або безпосередньо перенести настройки (і те, і інше робиться з правами root):

```
root@boincserver:/home/boincadm/projects/ toyota >
cat myapp.httpd.conf >> /etc/apache2/httpd.conf
```

Теперь нужно перезапустить сервер Apache:

```
root@boincserver:/home/boincadm/projects/ toyota >
apache2ctl restart
```

Якщо все налаштовано вірно, то при введенні в браузер адреси `http:// localhost / toyota` (з комп'ютера, на якому знаходиться проєкт) або `http://boincserver.domain.ru/ toyota` (з клієнтського комп'ютера).

Після того як була створена головна сторінка проєкту BOINC, яка є основною контактною точкою з майбутніми користувачами сервера, необхідно забезпечити запуск періодично виконуваних допоміжних служб (таких, наприклад, як служба освоєння результатів - `assimilator`). Робиться це

за допомогою системної служби cron: щоб встановити необхідну періодичність, досить виконати команду

```
boincadm @ boincserver: ~ / projects / myapp /> crontab toyota.cronjob
```

У файлі toyota.cronjob записано вказівку службі cron на запуск скрипта ~ / projects / toyota / bin / start кожні п'ять хвилин:

```
boincadm @ boincserver: ~ / projects / toyota />
more toyota.cronjob
0,5,10,15,20,25,30,35,40,45,50,55 * * * * / home
/ boinc / projects / toyota / bin / start -cron
```

Цей скрипт викликає все утиліти сервера BOINC, призначені для періодичного виконання.

Таким чином на цьому етапі виконані базові настройки сервера і залишається тільки сформулювати готову задачу, яку адміністратор задає в браузері.

4.3 Структура сценаріїв Python ¶

Каталог boinc / py / Boinc містить модуль Boinc. Це означає, що якщо є boinc / py / та користувач працює з python, можна написати:

```
from Boinc.setup_project import *
```

Щоб переконатися, що boinc / py / знаходиться в одному каталозі з python:

```
import boinc_path_config
```

Це спеціальний модуль, який налаштовує місця у відповідних каталогах, а потім відповідно змінює sys.path.

Модуль `boinc_project_path` імпортується, щоб отримати шляхи для `config.xml` та `run_state.xml`. Шляхи за замовчуванням для них - це батьківський каталог сценарію виклику.

```
boinc_project_path.py
config_xml_filename='/etc/boinc/home/config.xml'
run__state__xml__filename='/var/lib/boinc/home/r
un_state.xml'
```

Детальніше у файлі `boinc / py / Voinc / boinc_project_path.py`.

`Database.py` визначає внутрішню бібліотеку бази даних та взаємозв'язок між таблицями та об'єктами баз даних, щоб забезпечити легке керування даними.

Усі таблиці бази даних мають відповідний клас, а його рядки мають класи, де кожен стовпець є членом цього класу. Ідентифікатори автоматично перекладаються на і з об'єктів. Для початку імпортуйте модуль бази даних: `from Voinc import database`

Підключіться до бази даних: `database.connect_default_config()`

Класи таблиць можна індексувати за допомогою оператора `[]` для отримання об'єкта за ідентифікатором

```
# executes 'select * from project where id=1'.
# exception is raised if project is not found
project_with_id_1 = database.Projects[1]
```

Класи таблиць мають функцію пошуку, яка створює та виконує запит MySQL на основі його аргументів:

```
# this could return any number (0, 1, 2, ...) of
platforms
```

```
# executes "select * from platform where
user_friendly_name='commodore 64'"
list_of_platforms_called_c64=database.Platforms.
find(
    user_friendly_name = 'Commodore 64')
```

Об'єкти (рядки таблиці) мають дані своїх стовпців як члени, тому ви можете безпосередньо отримати до них доступ та змінити їх.

```
user_quar1=database.users.find(email_addr='quar1
@quar1.org')[0]
print 'name =', user_quar1.name
user_quar1.postal_code = 97404
```

Щоб створити новий об'єкт бази даних, створіть об'єкт Python і надайте всі значення як параметри ініціалізатору:

```
new_app = database.App(name=' toyota @home',
min_version=1, create_time=time.time())
```

Щоб зафіксувати будь-які зміни (включаючи новий об'єкт), викличте `commit()` (інструмент `boinc / tools / add.py` є інтерфейсом командного рядка для цього):

```
user_quar1.commit() # executes an UPDATE
new_app.commit()    # executes an INSERT
To remove an object, call remove():
team_eric_test=database.Teams(name="Eric's Test
Team")[0]
team_eric_test.remove()
# OR
```

```

    for team in database.Teams(name="Eric's Test
Team"):
        team.remove()
    #    OR
    map(database.Team.remove,database.Teams(name="Er
ic's Test Team"))

```

Щоб отримати доступ до об'єктів, пов'язаних за ідентифікатором, перейдіть до імені поля без суфікса «id»: (таблиця результатів має стовпці «workunitid» та «hostid»; таблиця хостів має ідентифікатор стовпця)

```

wu_1234=database.Workunits.find(name='1234.wu')[
0]
results_of_wu_1234=database.Results.find(workuni
t=wu_1234)
for result in results_of_wu_1234:
    os.system("echo 'you are crunching %s' | mail
'%s' " %(
        result.name, result.host.user.email_addr))

```

Після проведення налаштувань необхідно задати завдання використовуючи Voins та очікувати на виконання.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Налаштування LS-DYNA

З програмою LS-DYNA також встановлюється програмна оболонка ANSYS, яка дозволить запускати і використовувати необхідну програму.

На першому етапі установки необхідно запустити файл exe і погодитися з ліцензією (рис.5.1). Необхідно вибрати шлях установки файлів і вказати IP адреси для з'єднання з сервером ліцензій (рис.5.2). Після чого відбудеться запуск програми на ПК (рис. 5.3) для активації інструментів LS-DYNA (рис. 5.4).

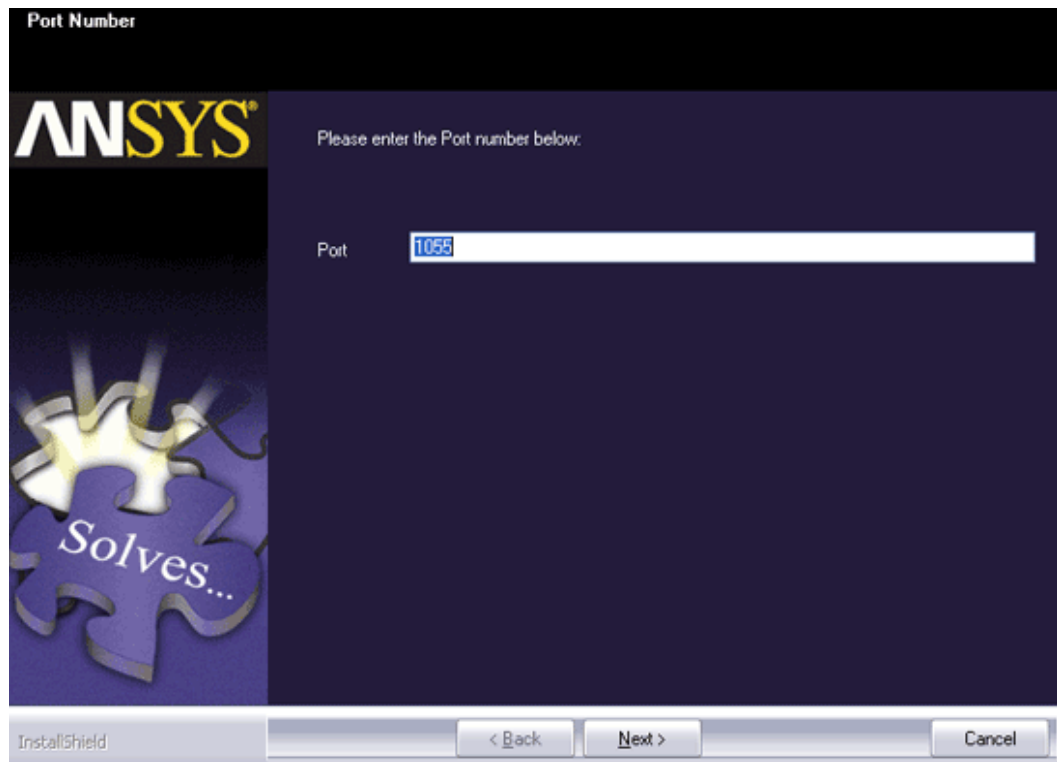


Рисунок 5.1 – Вибір порту сервера ліцензій

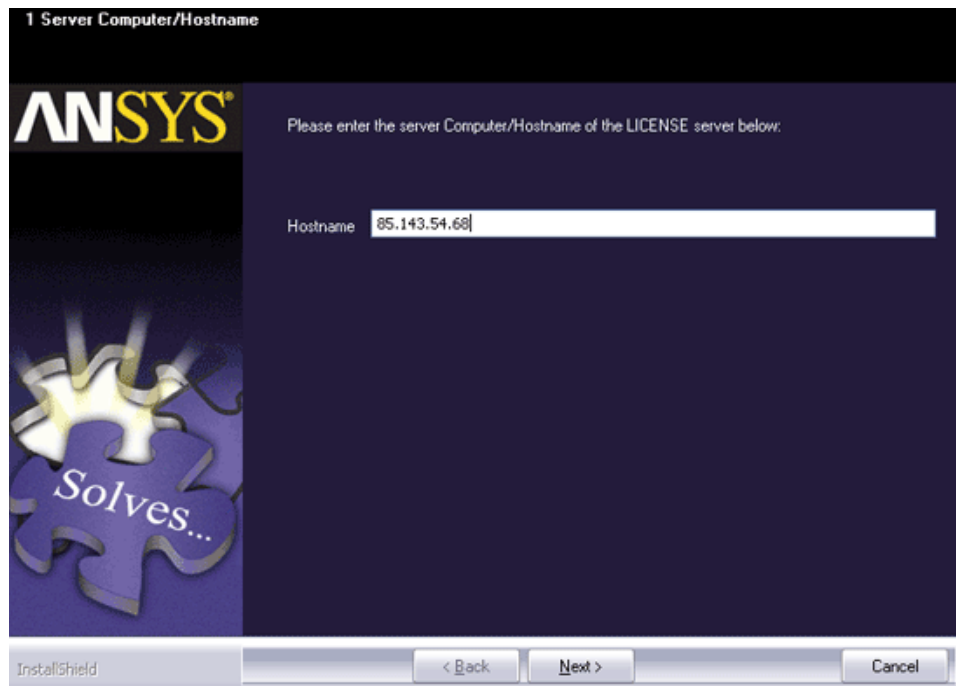


Рисунок 5.2 – Введення IP адреси сервера ліцензій

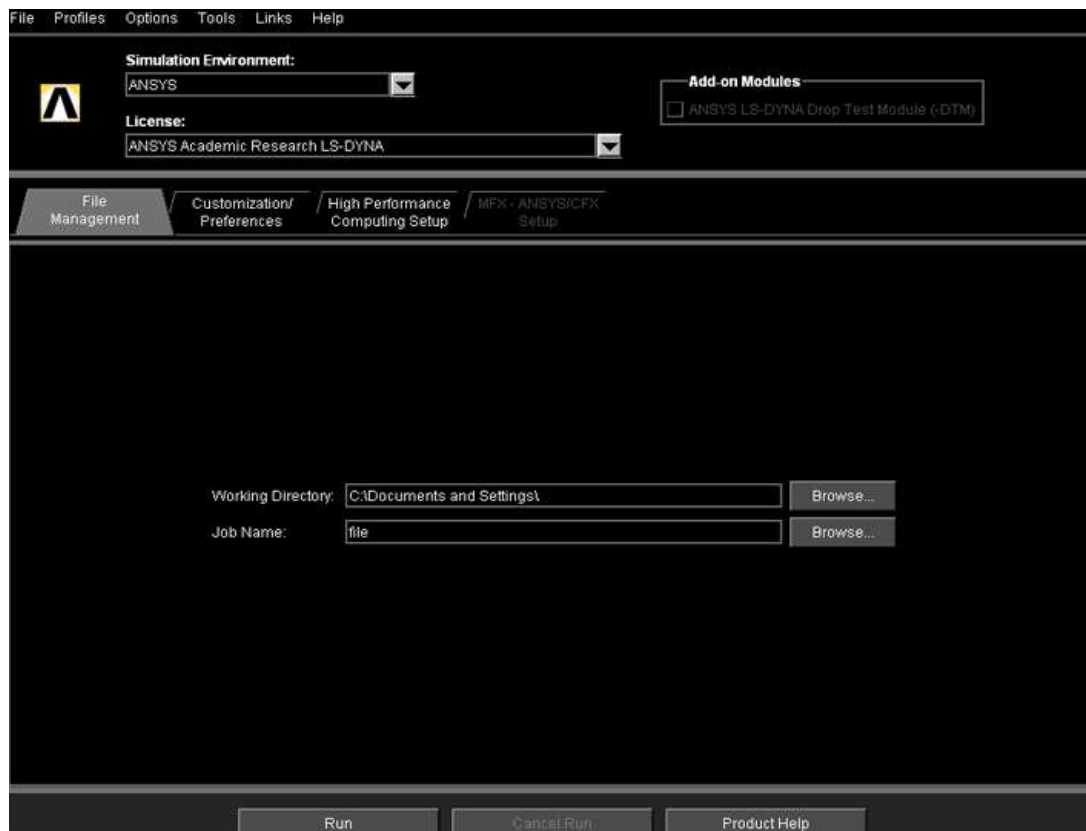


Рисунок 5.3 – Запуск на ПК

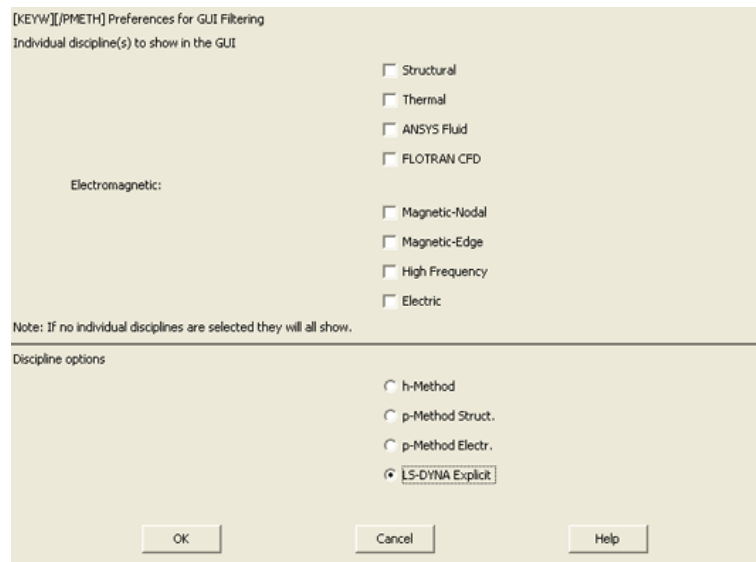


Рисунок 5.4 – Активація інструментів LS-DYNA

5.2 Налаштування Boinc-клієнта

Для установки клієнтської частини необхідно завантажити файл для установки з вільною ліцензією.

Після запуску даного файлу почнеться установка, після установки буде запропоновано запустити дану програму.

Після запуску необхідно вибрати проєкт якому користувач має намір брати участь (рис. 5.5).

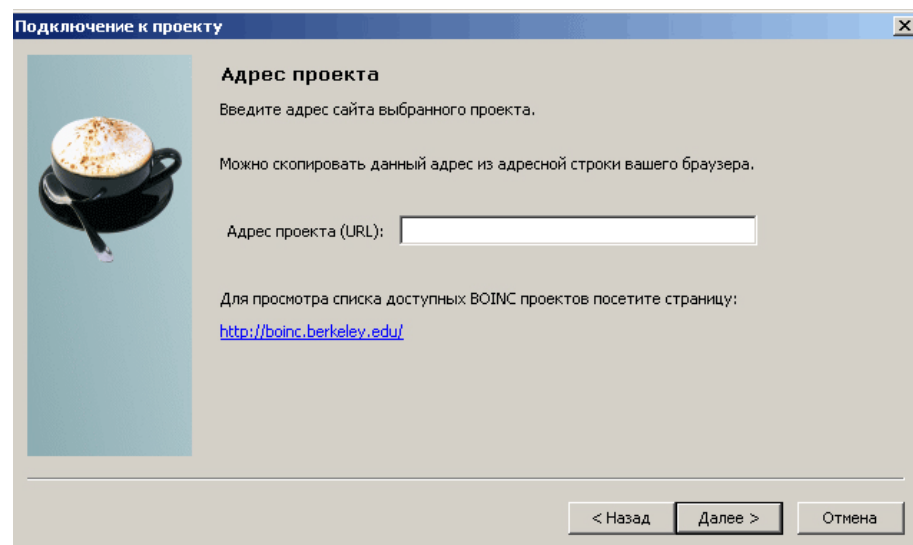


Рисунок 5.5 – Вибір проєкту

Після вибору проєкту необхідно налаштувати які ресурси користувач готовий надати. В даному меню користувач вказує коли проєкту можна споживати обчислювальну потужність, скільки процесорів можна використовувати, і на скільки відсотків можна їх використовувати (рис. 5.6).

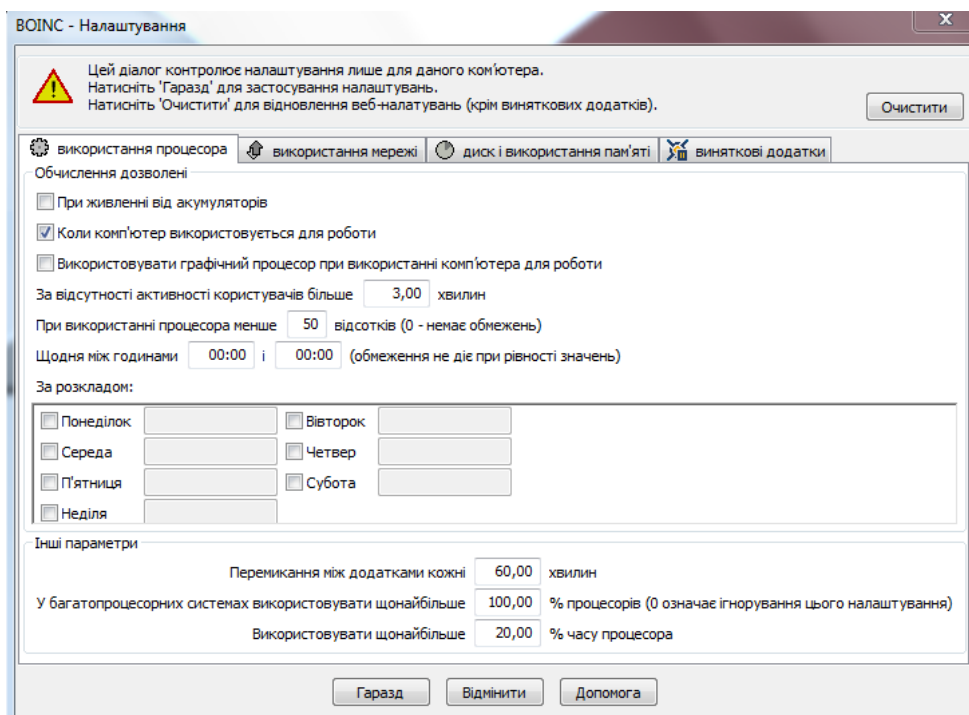


Рисунок 5.6 – Налаштування

Після даних налаштувань клієнт отримує завдання, які обчислюються на даних машинах.

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

У дипломній кваліфікаційній роботі магістра проведено дослідження та розроблено засоби оброблення даних для аналізу результатів випробувань технічних об'єктів.

Очікується, що впровадження в експлуатацію розробленого програмного забезпечення на підприємстві дозволить скоротити 0,8 ставки одного інженера з технічного діагностування, з заробітною платою 12800 гривень на місяць.

У розробці беруть участь програміст із заробітною платою 14800 грн. в місяць і консультант з заробітною платою 9800 грн. у місяць. На розробку проєкту було витрачено 2 місяці. Розробка програми розпочалася першого вересня і повинна бути виконана до кінця жовтня 2020 року.

6.1 Планування розробки програмного продукту

Весь комплекс розробки програмного продукту можна розділити на етапи [15]. Для кожного етапу вказуються трудомісткість, кількість виконавців і тривалість робіт. Інформація з характеристики робіт по розробці програмного забезпечення зведена у таблицю 6.1.

Таблиця 6.1 – Характеристика робіт по розробці програми

Найменування робіт	Трудомісткість		Виконавці	Тривалість, днів
	люд-дн	%		
Аналіз ПЗ	9	13,4	Програміст	5
			Консультант	4
Визначення вимог до ПП	7	10,4	Програміст	4
			Консультант	3

Продовження таблиці 6.1

Найменування робіт	Трудомісткість		Виконавці	Тривалість, днів
	люд-дн	%		
Проектування структури програми	18	26,9	Програміст Консультант	10 8
Розробка схеми функціонування програми	3	4,5	Програміст Консультант	2 1
Створення програмного коду	16	23,9	Програміст	16
Тестування і налагодження програмного продукту	9	13,4	Програміст Консультант	5 4
Складання програмної документації	5	7,5	Програміст Консультант	3 2
Разом	67	100	Програміст Консультант	45 22

6.2 Визначення витрат на розробку програми

Для визначення витрат на розробку програми складають калькуляцію кошторисної вартості робіт, що включає статті:

- основна заробітна плата;
- додаткова заробітна плата;
- відрахування на єдиний соціальний внесок (ЄСВ);
- матеріали та комплектуючі вироби;
- витрати на спеціальне обладнання (на оплату машинного часу

ЕОМ для написання і налагодження програмних засобів);

– накладні витрати.

Витрати за статтею «Основна заробітна плата» складаються з планового фонду зарплати всіх категорій працівників, зайнятих в розробці програми. Розрахунок зарплати на підставі даних про трудомісткість, представлений в таблиці 6.2.

Таблиця 6.2 – Основна заробітна плата

Посада виконавця	Чисельність, чол	місячний оклад	Кількість місяців роботи	Сума ЗП, грн
Програміст	1	14800	2	29600
Консультант	1	9800	1	9800
Разом	2			39400

Додаткову заробітну плату приймають рівною 10% від основної зарплати. Вона розраховується за формулою (6.1):

$$ЗП_{\text{дод}} = ЗП_{\text{осн}} \cdot 10\%. \quad (6.1)$$

Підставивши величину основної заробітної плати в дану формулу отримаємо: $ЗП_{\text{дод}} = 39400 \cdot 0.1 = 3940$ грн.

Відрахування на єдиний соціальний внесок (ЄСВ) визначають у відсотковому відношенні від сум основної та додаткової зарплати з урахуванням премій і доплат.

Відрахування вираховуються за формулою (6.2):

$$\begin{aligned} \text{ЄСВ} &= (ЗП_{\text{осн}} + ЗП_{\text{дод}}) \cdot 0,22, \\ \text{ЄСВ} &= (39400 + 3940) \cdot 0,22 = 9535 \text{ грн.} \end{aligned} \quad (6.2)$$

Використовується два найменування матеріалів: папір – 200 грн., картридж – 900 грн. Витрати на матеріали розраховують за формулою (6.3):

$$M = \sum_{i=1}^n (C_i \cdot N_i \cdot (1 + K_{m.з.}) - C_{io} \cdot N_{io}), \quad (6.3)$$

де M – витрати на матеріали, покупні напівфабрикати і комплектуючі вироби, грн .;

$K_{m.з.}$ – коефіцієнт, що враховує транспортно-заготівельні витрати;

C_i – ціна і-го найменування матеріалу, напівфабрикату і комплектуючого, грн .;

N_i – потреба в і-му матеріалі, напівфабрикаті і комплектуючому;

C_{io} – ціна зворотних відходів і-го найменування матеріалу, грн;

N_{io} – кількість зворотних відходів і-го найменування;

n – кількість найменувань матеріалів, напівфабрикатів і комплектуючих.

$$C_{io} = 0; N_{io} = 0; K_{m.з.} = 0,05,$$

$$M = (1 + 0,05) \cdot (100 + 900) = 1155 \text{ грн.}$$

Таким чином, разом витрати на матеріали та комплектуючі вироби становлять 1155 гривень.

У статтю «Витрати на спеціальне обладнання» входять витрати на придбання, транспортування, монтаж і налагодження нестандартного обладнання. Практично, в даному випадку, в цій статті враховуються витрати на оплату машинного часу ЕОМ для написання і налагодження програмних засобів. Для цього необхідно скласти кошторис «витрат на утримання і експлуатацію устаткування», виходячи з якого визначиться вартість одного машино-години роботи ПК, після множення якої на машинний час пішло на

написання і налагодження програми отримаємо витрати на оплату машинного часу.

Амортизаційні відрахування визначають за формулою (6.4):

$$A = BB \cdot \frac{H_a}{100}, \quad (6.4)$$

де BB – балансова вартість обчислювальної техніки, грн;

H_a – норма амортизаційних відрахувань на повне оновлення обчислювальної техніки,%. Приймаємо, що техніка повинна працювати 4 роки, тому $H_a = 25\%$.

Балансова вартість обчислювальної техніки становить 27400 грн. Тому амортизаційні відрахування становлять:

$$A = 27400 \cdot 0,25 = 6850 \text{ грн.}$$

Статтю "Експлуатація обладнання" розраховують підсумовуванням витрат на силову електроенергію і допоміжні матеріали. Витрати на електроенергію визначають за формулою (6.5):

$$C_{\varepsilon} = N_n \cdot \Phi_{\varepsilon\phi} \cdot K_{\varepsilon\phi} \cdot K_{\varepsilon\phi} \cdot K_{\varepsilon\phi} \cdot C_{\varepsilon}, \quad (6.5)$$

де N_n – номінальна потужність ЕОМ, кВт;

$\Phi_{\varepsilon\phi}$ – річний ефективний фонд часу роботи ЕОМ, машино–год;

$K_{\varepsilon\phi}$ – середній коефіцієнт завантаження за часом;

$K_{\varepsilon\phi}$ – коефіцієнт завантаження по потужності;

C_{ε} – ціна одного кВт · год електроенергії, грн ./ (кВт · год).

Номінальна Потужність ЕОМ – 0,2 кВт. Річний ефективний фонд часу роботи ЕОМ ставити 1800 годин. Середні коефіцієнти завантаження за часом

та за потужністю рівні відповідно 0,9 та 0,6. Ціна однієї кіловат-години електроенергії ставить 2,11 грн. Отримуємо:

$$C_e = 0,2 \cdot 1800 \cdot 0,9 \cdot 0,6 \cdot 2,11 = 410,18 \text{ грн.}$$

Приймаємо, що $C_e = 410$ грн.

Витрати за статтею "Заробітна плата обслуговуючих робітників та відрахування на соціальне страхування в інші фонди" визначають за формулою (6.6):

$$ЗП_{обсл} = ФЗП_z \cdot \left(1 + K_{відрах.}\right) \cdot \frac{t_{обсл}}{\Phi_{ef.обсл}}, \quad (6.6)$$

де $ФЗП_z$ – річний фонд заробітної плати (основної та додаткової) обслуговуючих робітників, грн.;

$K_{відрах.}$ – коефіцієнт, що враховує відрахування на соціальне страхування і в інші фонди;

$t_{обсл}$ – час протягом року, необхідне технічне обслуговування ЕОМ, годин / рік. Приймаємо $t_{обсл} = 36$ год.;

$\Phi_{ef.обсл}$ – річний ефективний фонд часу обслуговуючого персоналу, годин / рік. Приймаємо, що річний ефективний фонд часу обслуговуючого персоналу становить 1800 годин

Заробітна плата обслуговуючого персоналу = 8900 грн/місяць.

$$ФЗП_z = 8900 \cdot 12 = 106800 \text{ грн.,}$$

$$ЗП_{обсл} = 106800 \cdot (1 + 0,22) \cdot 36 / 1800 = 2606 \text{ грн.}$$

Суму витрат за статтею "Поточний ремонт обладнання" обчислимо як 3% від балансової вартості обладнання, оскільки неможливо заздалегідь

визначити, скільки коштів необхідно витратити на придбання запасних частин наявного обладнання. Балансова вартість обчислювальної техніки становить 27400 грн. Тому сума витрат за статтею "Поточний ремонт обладнання" становить: $27400 \cdot 0,03 = 822$ грн.

Інших витрат за проектом не передбачено, тому сума витрат за цією статтею дорівнює нуль гривень.

Кошторис витрат на утримання і експлуатацію устаткування наведений в таблиці 6.3.

Таблиця 6.3 – Кошторис витрат на утримання і експлуатацію устаткування

Найменування статей витрат	Сума, грн.
Амортизація обладнання	6850
Заробітна плата основна і додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	2606
Експлуатація обладнання (крім витрат на поточний ремонт)	410
Поточний ремонт обладнання	822
Інші витрати	0
Разом	10688

Витрати на оплату машинного часу ЕОМ для написання і налагодження програмних засобів визначаються за формулою (6.7):

$$C_{mo} = P_{екс} \cdot t_{mo}, \quad (6.7)$$

де C_{mo} – витрати на оплату машинного часу, грн;

$P_{екс}$ – експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн. / машино–год.;

$t_{мо}$ – машинний час цифрової ЕОМ для написання і налагодження даного програмного продукту, машино–год.

$$P_{екс} = 10688/1800 = 5,94 \text{ грн. / машино–год.}$$

Для написання і налагодження даного програмного продукту ЕОМ експлуатується протягом 45 днів в одну зміну по 8 годин, що складає в сумі 360 годин. Таким чином отримуємо суму за статтею «Витрати на спеціальне обладнання»:

$$C_{мо} = 5,94 \cdot 360 = 2138 \text{ грн.}$$

До накладних витрат відносяться витрати на загальне управління і загальногосподарські потреби (заробітна плата апарату управління, канцелярські витрати і т.д.), утримання та експлуатацію будівель. Накладні витрати включаються до вартості розробки програми непрямым шляхом – у відсотках до основної заробітної плати розробників. Для обчислення цієї статті видатків у дипломному проєкті приймемо, що накладні видатки становлять 15% від основної заробітної плати розробників, що складає 39400 грн. Тому сума накладних видатків становить: $39400 \cdot 0,25 = 9850$ грн.

Калькуляція кошторисної вартості робіт з розробки програми наведена в таблиці 6.4.

Таблиця 6.4 – Калькуляція кошторисної вартості робіт з розробки програми

Найменування статей витрат	Сума, грн.
Основна заробітна плата	39400

Продовження таблиці 6.4

Найменування статей витрат	Сума, грн.
Додаткова заробітна плата	3940
ЄСВ	9535
Матеріали і комплектуючі	1155
Витрати на спеціальне обладнання	2138
Накладні витрати	5910
Разом	62078

6.3 Розрахунок економічної ефективності програмного продукту

Річну економію від впровадження програми, що дозволяє знизити витрати машинного часу ЕОМ для рішення певного завдання в порівнянні із програмою, що застосовувалася раніше, визначають за формулою (6.8):

$$E = (T_{M_1} - T_{M_2}) \cdot B_{\text{екс.КТЗ}} + E_{\text{н}}^a \cdot \frac{\Phi_{\text{б.КТЗ}} \cdot (T_{M_1} - T_{M_2})}{\Phi_{\text{еф.КТЗ}}} - \frac{S_{\text{рп}}}{T_{\text{с}}} + E_{\text{зп}}, \quad (6.8)$$

де T_{M_1}, T_{M_2} – машинний час, необхідний для розв’язання поставлених завдань, відповідно в старому та у новому варіантах, машино–год./рік;

$B_{\text{екс}}$ – експлуатаційні витрати, що доводяться на 1 год машинного часу ЕОМ, грн/ машино–год.;

$E_{\text{н}}^a$ – нормативний коефіцієнт ефективності капітальних вкладень у засоби автоматизації;

$\Phi_{\text{б.КТЗ}}$ – балансова вартість ЕОМ (КТЗ), грн.;

$\Phi_{\text{еф.КТЗ}}$ – річний ефективний фонд часу роботи ЕОМ, машино–год.;

$S_{\text{рп}}$ – сумарні витрати на розробку програми, грн.;

T_c – термін служби впроваджуваної програми до її морального зношування, років. Приймаємо цей термін 5 років;

$E_{зп}$ – економія на заробітній платі.

До впровадження даного ПЗ дані функції виконувались співробітником, що відповідає за обробку рукописних символів, 1800 годин на рік, а після 360 годин на рік (економія складає 0,8 ставки одного інженера з технічного діагностування, з заробітною платою 12800 гривень на місяць).

$$E = (1800 - 360) \cdot 5,94 + 0,2 \cdot (27400 \cdot (1800 - 360)) / 1800 - 62078/5 + \\ + 12800 \cdot 0,8(1 + 0,22) \cdot 12 = 150436 \text{ грн.}$$

Коефіцієнт ефективності капітальних вкладень E и строк їхньої окупності $T_{ок}$ розраховуються за формулами (6.9) і (6.10):

$$Ef = \frac{E}{S_{рп}}, \quad (6.9)$$

$$T_{ок} = \frac{S_{рп}}{E}, \quad (6.10)$$

$$Ef = 150436 / 62078 = 2,42,$$

$$T_{ок} = 62078 / 150436 = 0,41 \text{ року.}$$

Зведемо отримані техніко–економічні показники в таблицю 6.5.

Таблиця 6.5 – Техніко–економічні показники

Показники	Сума	Одиниця виміру
Витрати на розробку програми	62078	грн.
Річна економія	150436	грн.
Строк окупності програми	0,41	рік

Таким чином, впровадження в експлуатацію розробленого

програмного забезпечення на виробничому підприємстві дозволить скоротити 0,8 ставки одного інженера з технічного діагностування, з заробітною платою 12800 гривень на місяць. Економічні розрахунки показали, що строк окупності програми 0,41 року, що значно менше часу її нормативного строку окупності (5 років). Крім того, річна економія при використанні програми становить 150436 грн. Виходячи з виконаних розрахунків, можна зробити висновок, що розробка даного програмного продукту є економічно доцільною.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

У даному розділі розглянуто основні потенційні небезпеки приміщення, де безпосередньо проводиться дослідження та розробка засобів оброблення даних для аналізу результатів випробувань технічних об'єктів.

Основна робота при розробці проводиться за комп'ютером, тому проаналізовано основні фактори, що так чи інакше впливають на стан здоров'я дослідника:

- незадовільні параметри мікроклімату робочої зони (значення показників мікроклімату у виробничих приміщеннях);
- відсутність або недостача природного освітлення;
- недостатня освітленість робочого місця;
- широкий спектр випромінювання від дисплея, який включає рентгенівську, ультрафіолетову і інфрачервону області, а так само широкий діапазон електромагнітних випромінювань інших частот;
- небезпека виникнення пожежі;
- статична напруга, яка здатна вивести з ладу електронні компоненти;
- небезпечний рівень напруги який може видавати блок живлення ПК;
- погано організоване робоче місце, що може привести до уражень струмом, механічних травм та інше.

Розробка проводиться в приміщенні, що має розміри 25x15м². Також у цьому приміщенні розташовано 24 робочих місця, обладнаних персональними комп'ютерами.

Слід відзначити, що приміщення має 3 досить великих вікна, але оскільки вони розташовані з одного боку та не по центру, то в приміщенні може виникнути проблема з освітленням. Також приміщення обладнане великою кількістю розеток, що мають 3 контакти (фаза, нуль та заземлення).

Для забезпечення відповідного мікроклімату приміщення обладнано двома спліт-системами кондиціонування. Вони ж і забезпечують іонізацію повітря та його зволоження.

7.2 Заходи із забезпечення безпеки

Робота з обчислювальною технікою зумовлює виникнення фактору небезпеки, що пов'язаний з електрострумом.

З метою попередження уражень електричним струмом до роботи допускаються лише особи, які добре освоїли основні правила техніки безпеки.

Для запобігання ураження людини, при дотику до частин ЕОМ, що знаходяться під напругою, виконується захисне заземлення. Суть захисного заземлення - зменшення напруги на корпусі електроустаткування при пробії фази на корпус до безпечної величини. Сучасні ЕОМ включаються в розетку за допомогою триполюсною вилки, один з полюсів якої використовується для заземлення. Згідно ПУЕ опір захисного заземлення в приміщенні для комп'ютерної техніки не перевищує 4 Ом, що значно менше опору тіла людини, тому небезпека ураження при дотику до несправного електроустаткування значно зменшується, так як струм в цьому випадку піде шляхом найменшого опору [16].

Специфічна небезпека електроустановок - дроти, за якими проходить струм, корпуси стійок ЕОМ і іншого устаткування. Реакція людини на електричний струм виникає лише при протіканні струму через тіло людини.

Дуже важливе значення для запобігання електротравматизму має правильна організація обслуговування електроустановок, проведення ремонтних, монтажних і профілактичних робіт, а також організація правильного заземлення обладнання.

При організації заземлення зв'язок корпусу ПК з робочим нулем мережі живлення неприпустимий, оскільки при обриві робочого нуля корпус ПК

виявляється під «фазною» напругою, крім того, при великому навантаженні в електричній мережі на робочому нулі зосереджується значний потенціал.

Для правильного підключення обов'язкова наявність захисного нульового (третього) дроту мережі і зв'язок з ним відповідного проводу ПК за допомогою спеціального контакту стандартної трехпроводної розетки. Таке підключення використано для забезпечення електробезпеки обладнання.

Для забезпечення безпеки робіт в електроустановках, які діють, виконуються наступні організаційні заходи:

- а) призначення осіб, відповідальних за організацію і виконання робіт;
- б) оформлення робіт нарядом або усним розпорядженням;
- в) допуск до роботи;
- г) спостереження під час проведення робіт;
- д) забезпечення перерви в роботі;
- е) при проведенні робіт зі зняттям напруги в електроустановках, які діють, або поблизу них, необхідно:

- 1) відключення установки від джерела живлення електроенергією;
- 2) механічне замикання приводів відключених комутаційних апаратів;
- 3) зняття запобіжників;
- 4) від'єднання кінців ліній живлення;
- 5) заходи, які забезпечують неможливість помилкової подачі напруги до місця роботи;
- 6) використання заземлень [17].

Окрім ураження електрострумом при розробці можна отримати механічні пошкодження від інструментів, що необхідні для розробки мережі.

А саме, для роботи з кабелем (витою парою) необхідно використовувати спеціальний інструмент – кримпер, що має гострі леза для перерізання кабелів та має спеціальний механізм для стискання конекторів. Тому, це досить травмонебезпечний пристрій, який може спричинити

механічні пошкодження частин тіла людини. Тому при роботі з ним необхідно дотримуватись таких правил безпеки:

- ознайомитись з інструкцією приладу;
- одягнути спеціальні захисні перчатки для захисту рук;
- переконатися у тому, що робоче місце підходить для роботи з інструментом;
- переконатись, що інструмент справний;
- після роботи з інструментом, закрити гострі елементи спеціальним захисним ковпачком [18].

7.3 Заходи з виробничої санітарії та гігієни праці

У приміщеннях, де встановлені персональні комп'ютери, дотримані визначені параметри мікроклімату. У санітарних нормах ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень [19] установлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються в залежності від часу року, характеру трудового процесу і характеру виробничого приміщення.

У виробничому приміщенні застосовується водяна система опалення, яка рівномірно прогріває повітря, при цьому відсутнє виділення запахів гару, таке опалення відповідає основним санітарно-гігієнічним вимогам.

Контроль за станом виробничого та навколишнього середовища, за відповідністю характеристик робочих місць нормативними актами здійснюють фахівці лабораторії охорони навколишнього середовища і промислової санітарії.

Робота у приміщенні пов'язана з розумовою працею. Тому для попередження стомлення і підвищення працездатності цих осіб визначено найбільш раціональний режим роботи і відпочинку.

На працівників впливає рівень шумів.

Допустимий рівень шуму в приміщенні становить в діапазоні 50 - 71 дБ, що відповідає вимогам ДСН 3.3.6-037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» [20].

На робочих місцях в приміщенні шум створюється технічними засобами, від зовнішніх джерел шуму (вентиляторів, кондиціонерів) і ін.

Під дією шуму у людини знижується гострота зору, слуху, підвищується кров'яний тиск, ослаблення уваги, пам'яті.

Зниження шуму здійснюється наступними методами: зменшенням шуму в джерелі; раціональним плануванням приміщення; акустичною обробкою приміщення. Відповідно, зменшення шуму у приміщенні для роботи досягається збільшенням звукоізоляції конструкцій, ущільненням по периметру притворів вікон, дверей [21].

Окрім цього на людину впливає якість освітлення. При проектуванні освітлювальної установки вирішуються такі питання:

- вибір типу джерела світла;
- визначення системи освітлення;
- вибір типу світильників з урахуванням умов середовища;
- розподіл світильників і визначення їх кількості;
- визначення норми освітленості на робочому місці.

У приміщенні, де виконується робота, застосовується сумісне освітлення (природне і штучне). Найбільш прийнятним для подібних приміщень є використання люмінесцентних ламп білого світла.

Згідно ДБН В.2.5-28:2018 «Природне і штучне освітлення» [22] для найменшого розрізнення об'єктів (розряд зорової роботи III (б)) 0,3–0,5 мм значення коефіцієнта природного освітлення (КПО) становить 1,5%.

Згідно з ДБН В.2.5-28-2006 рівень мінімального нормованого загального освітлення в приміщенні становить не менше $E_n = 200$ лк.

Для забезпечення нормальних санітарно-гігієнічних умов у відповідності до вимог ДБН В.2.-67:2013 «Опалення, вентиляція та кондиціонування» [23] в приміщеннях передбачається загально обмінна

припливно-витяжна механічна вентиляція і кондиціонування повітря. Основним завданням кондиціонування повітря є підтримка параметрів повітряного середовища в допустимих межах, що забезпечують надійну роботу ЕОМ, тривале зберігання носіїв інформації та комфортні умови для обслуговуючого персоналу. Кондиціонування здійснюється побутовими кондиціонерами, встановленими в вікнах. Окрім цього приміщення обладнане вентиляцією.

Кількість вентиляційного повітря визначається за формулою (7.1) [24]:

$$V_{\text{вент}} = \frac{3600 \cdot Q_{\text{надл}}}{C \cdot \rho \cdot (t_{\text{вид}} - t_{\text{пр}})}, \quad (7.1)$$

де $Q_{\text{надл}}$ – виділення в приміщенні явного тепла, Bt ;

C – теплоємність повітря ($C = 1000 \text{ Дж} / \text{кг} \cdot \text{гр}$);

ρ – питома щільність повітря ($\rho = 1,3 \text{ кг} / \text{м}^3$);

$t_{\text{вид}}$ і $t_{\text{пр}}$ – температури повітря, що видаляється та приточного, $^{\circ}\text{C}$.

Температура повітря, що видаляється визначається з формули (7.2):

$$t_{\text{вид}} = t_{\text{зн}} + d(h - 2), \quad (7.2)$$

де $t_{\text{зн}}$ – температура повітря в зоні праці ($t_{\text{зн}} = 21^{\circ}\text{C}$);

d – коефіцієнт наростання температури на кожен метр висоти ($d = 1,5^{\circ}\text{C} / \text{м}$);

h – висота приміщення ($h = 5 \text{ м}$).

Звідси $t_{\text{вид}} = 21 + 1,5(5 - 2) = 25,5 (^{\circ}\text{C})$.

Кількість надлишкового тепла визначається з теплового балансу, як різниця між теплом, що надходить у приміщення і теплом, що видаляється з приміщення і що поглинається в ньому.

$$Q_{\text{надл}} = Q_{\text{надх}} - Q_{\text{випр}}, \quad (7.3)$$

Тепло, що надходить у приміщення, визначається за формулою:

$$Q_{\text{надх}} = Q_{\text{устатк}} + Q_{\text{л}} + Q_{\text{осв}} + Q_{\text{рад}}, \quad (7.4)$$

де $Q_{\text{устатк}}$ – тепло від роботи устаткування;

$Q_{\text{л}}$ – тепло, що надходить від людей;

$Q_{\text{осв}}$ – тепло від джерел освітлення;

$Q_{\text{рад}}$ – тепло від сонячної радіації, що надходить через вікна.

$$Q_{\text{устатк}} = E_{\text{устатк}} \cdot P_{\text{уст}} = 0,15 \cdot 750 = 112,5 \text{ (Вт)}, \quad (7.5)$$

де $E_{\text{уст}}$ – частка енергії, що переходить у тепло;

$P_{\text{уст}}$ – потужність установки.

$$Q_{\text{л}} = n \cdot q = 10 \cdot 90 = 900 \text{ (Вт)}, \quad (7.6)$$

де, n – кількість осіб у залі ($n = 10$);

q – кількість тепла, що виділяється людиною ($q = 90 \text{ Вт}$).

$$Q_{\text{осв}} = E_{\text{осв}} \cdot P_{\text{осв}} = 0,4 \cdot 305 = 122 \text{ (Вт)}, \quad (7.7)$$

де, $E_{\text{л}} = 0,4$ для люмінесцентних ламп;

$P_{\text{осв}}$ – потужність освітлювальної установки.

$$Q_{\text{рад}} = A \cdot k \cdot S \cdot t = 180 \cdot 0,8 \cdot 4,5 \cdot 3 = 1944 \text{ (Вт)}, \quad (7.8)$$

де, A – теплонадходження в приміщення з 1 м² скла ($127 - 234 \text{ Вт} / \text{м}$);

S – площа вікна ($S = 4,5 \text{ м}^2$);

m – кількість вікон ($m = 3$);

k – коефіцієнт, що враховує характер скління ($k = 0,8$).

З формули (7.4) отримуємо :

$$Q_{\text{надх}} = 112,5 + 900 + 122 + 1944 \approx 3078,5 \text{ (Вт)}.$$

$$Q_{\text{випр}} = 0,1 \cdot Q_{\text{надх}} = 0,1 \cdot 3078,5 = 307,85 \text{ (Вт)}. \quad (7.9)$$

Звідси за формулою (7.3) знаходимо:

$$Q_{\text{надл}} = 3078,5 - 307,85 \approx 2770,65 \text{ (Вт)}.$$

Знаходимо необхідний повітрообмін:

$$V_{\text{вент}} = \frac{3600 \cdot 2770,65}{1000 \cdot 1,3 \cdot (25,5 - 18)} = \frac{9974340}{9750} \approx 1023 \text{ (м}^3 / \text{год)}$$

Оптимальним варіантом є кондиціонування повітря, тобто автоматична підтримка його стану в приміщенні відповідно до визначених вимог (задана температура, вологість, рухливість повітря) незалежно від зміни стану зовнішнього повітря й умов у самому приміщенні.

Визначаємо необхідну кратність повітрообміну:

$$K = \frac{V_{\text{вент}}}{V_{\text{прим}}}, \quad (7.10)$$

де,

$$V_{\text{прим}} = n \cdot S_{\text{ос}} \cdot h, \quad (7.11)$$

n – число осіб у приміщенні;

S_{oc} – площа виробничого приміщення, що приходить на 1 особу (за нормами для розумово-фізичної праці $S_{oc} = 6 \text{ м}^2$);

h – висота приміщення.

Кратність повітрообміну: $K = \frac{1023}{10 \cdot 6 \cdot 5} = 3.41$.

7.4 Заходи безпеки у надзвичайних ситуаціях

7.4.1 Заходи з пожежної безпеки

Заходи з протипожежної безпеки розроблені для виробничого приміщення обладнаного ПК з ВДТ розроблені згідно вимог НАПБ А.01.001-2014 «Правила пожежної безпеки в Україні» [25].

Виходячи з аналізу речовин та матеріалів, які використовуються при роботі у приміщенні, обладнаному ПК з ВДТ:

- згідно ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» [26] у приміщенні, обладнаному ПК з ВДТ можлива пожежа класу – А та Е; воно належить до категорії «Д» з пожежної небезпеки;

- простір у приміщенні, у якому перебувають тверді горючі речовини та матеріали.

Оскільки приміщення належить до категорії «Д» з пожежної небезпеки, тому відповідно до вимог ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» [26] воно має II ступінь вогнестійкості.

У разі виникнення пожежі у приміщенні обладнаному ПК з ВДТ для евакуації персоналу відповідно до вимог ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» передбачені виходи, по обидві сторони приміщення, з одного боку вікно (на пожежні сходи), а з

іншого – вхідні двері. Згідно СНиП 2.09.02-85* «Производственные здания», відстань від найбільш віддаленого робочого місця до найближчого евакуаційного виходу не обмежується.

Обладнання, силові та освітленні мережі приміщення, обладнаного ПК з ВДТ відповідають вимогам пожежної безпеки, оскільки виконані відповідно до вимог НПАОП 40.1-1.32-01 «Правила будови електроустановок. Електрообладнання спеціальних установок» [27], та мають ступінь захисту ізоляції обладнання IP44 яка відповідає класу пожежанебезпечної зони П-Па до якої належить приміщення.

Для виявлення початкової стадії загоряння, швидкого й точного оповіщення служби пожежної охорони про час і місце виникнення пожежі встановлені системи автоматичної пожежної сигналізації (АПС). Вони можуть самостійно пускати в хід установки пожежогасіння, коли пожежа не досягла великих розмірів. Системи АПС складаються з пожежних сповіщувачів, ліній зв'язку й прийомних пультав (станцій). Згідно вимог ДБН В.2.5-56:2014 «Системи протипожежного захисту», в приміщенні обладнаному ПК з ВДТ встановлена система пожежної й охоронної сигналізації «Фотон-ОФІС», яка забезпечує виявлення теплових і димових ознак пожежі і місця виникнення пожежі з точністю до місця розміщення датчика [25].

Оскільки приміщення що обладнане ПК з ВДТ має площу 375 м², тому відповідно до вимог п. 3.8 розділу «Типові норми належності вогнегасників» ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників. Загальні технічні вимоги» [28] для гасіння електроустановок, що знаходяться під напругою, передбачені вуглекислотні вогнегасники типу ВВК-3,5 у кількості 19 штук (з розрахунку один вогнегасник с величиною заряду вогнегасної речовини 3 кг. і більше, на 20 м² площі приміщення). Відстань між вогнегасниками та місцями можливих загорянь не перевищує 10 м.

7.4.2 Заходи з цивільного захисту

Дії виробничого персоналу об'єкта при можливих аваріях, катастрофах і стихійних лихах. Заходи з ліквідації наслідків і захисту персоналу.

Стихійні дії сил природи, поки ще не в повній мірі підвладні людині, наносять економіці держави і населенню величезний збиток. Стихійні лиха - такі явища природи, які викликають екстремальні ситуації, порушують нормальну життєдіяльність людей і роботу об'єктів.

Великі аварії і катастрофи на об'єктах можуть виникати в результаті стихійного лиха, а також порушення технології виробництва, правил експлуатації різних машин, обладнання та встановлених заходів безпеки. Їх впливу подібні стихійних лих.

Під аварією розуміють раптову зупинку роботи або порушення процесу виробництва на промисловому підприємстві, транспорті, інших об'єктах, що призводять до пошкодження або знищення матеріальних цінностей.

Під катастрофою розуміють раптове лихо; подія, що несе за собою трагічні наслідки. Катастрофи супроводжуються руйнуванням будівель, різних споруд, знищенням матеріальних цінностей і загибеллю людей.

Найбільш повне і організоване виконання заходів цивільного захисту (ЦЗ) на об'єкті досягається завчасної розробкою плану заходів, які необхідно провести при виникненні загрози.

При ліквідації наслідків стихійних лих, великих аварій і катастроф передбачається організація оповіщення робітників і службовців працюючої зміни, формувань і населення відомчих будинків про загрозу виникнення або виникли аварії та стихійні лиха. Зазвичай для цього використовуються мережі внутрішнього радіомовлення, телефонної і диспетчерського зв'язку. Передбачаються заходи щодо організації та проведення евакуації робітників, службовців з цехів (підрозділів) об'єкта, а населення - з відомчих будинків, вказуються маршрути і пункти евакуації, порядок видачі засобів індивідуального захисту особового складу формувань, робочим, службовцям

цехів (підрозділів). Визначаються сили і засоби для проведення робіт з ліквідації наслідків великих аварій і катастроф, порядок управління, сили і засоби зв'язку, щоб забезпечити управління. Крім того, передбачають організацію харчування особового складу формувань ЦЗ і постраждалого населення.

При загрозі для життя виробляють евакуацію персоналу. Для евакуації людей використовують основні та запасні входи і виходи, віконні прорізи і балкони, люки в перекриттях, іноді спеціально проробляють отвори в перегородках і стінах [29].

Ліквідацію аварій на водопровідних мережах роблять у два етапи. Спочатку їх локалізують, а потім відновлюють пошкоджені ділянки. Локалізація аварій дозволяє обмежити або запобігти розмиву ґрунту, доріг, затоплення підвальних і інших приміщень.

Серед захисних заходів цивільної оборони, здійснюваних завчасно, особливо важливе місце займає організація оповіщення органів цивільного захисту, формувань і населення про загрозу нападу противника і про застосування ним ядерної, хімічної, бактеріологічної (біологічної) зброї і інших сучасних засобів нападу. Особливе значення оповіщення стоїть у разі раптового нападу противника, коли реальний час для попередження населення буде вкрай обмеженим і обчислюватися хвилинами.

З метою своєчасного попередження населення міст і сільських населених пунктів про виникнення безпосередньої небезпеки застосування супротивником ядерного, хімічного, бактеріологічного (біологічного) або іншої зброї і необхідності застосування заходів захисту встановлені наступні сигнали оповіщення цивільної оборони: «Повітряна тривога» «Відбій повітряної тривоги», «Радіаційна небезпека»; «Хімічна тривога».

Сигнал «Повітряна тривога» подається для всього населення. Він попереджає про безпосередню небезпеку ураження противником даного (району). За радіотрансляційної мережі передається текст: «Увага! Увага! Громадяни! Повітряна тривога! Повітряна тривога!» Одночасно з цим сигнал

дублюється звуком сирен, гудками дзвінків і транспортних засобів. На об'єктах сигнал буде дублюватися усіма, наявними в їх розпорядженні засобами. Тривалість сигналу 2-3 хвилини.

За цим сигналом об'єкти припиняють роботу, транспорт зупиняється і все населення ховається в захисних спорудах. Робітники і службовці припиняють роботу відповідно до встановленої інструкцією і вказівками адміністрації, що виключають виникнення аварій.

Сигнал «Радіаційна небезпека» подається в населених пунктах і районах, у напрямку до яких рухається радіоактивна хмара, що утворилася при вибуху ядерного боєприпасу.

За сигналом «Радіаційна небезпека» необхідно надягнути респіратор, протипилову тканинну маску і ватна-марлеву пов'язку, а при їх відсутності - протигаз, узяти підготовлений запас продуктів, індивідуальні засоби медичного захисту, предмети першої необхідності і піти в притулок, протирадіаційне або найпростіше укриття.

Сигнал «Хімічна тривога» подається при загрозі або безпосередньому виявленні хімічного або бактеріологічного нападу (зараження). З цього приводу сигналу необхідно швидко надіти протигаз, а в разі необхідності - засоби захисту шкіри і при першій же можливості сховатися в захисній споруді.

Якщо захисної споруди поблизу не виявиться, то в разі ураження аерозолями отруйних речовин і бактеріальних засобів можна сховатися в житлових, виробничих або підсобних приміщеннях.

Якщо буде встановлено, що супротивник застосував бактеріологічна (біологічна) зброя, то по системам оповіщення населення отримає рекомендації про подальші дії.

Необхідно бути гранично уважними і строго виконувати розпорядження органів цивільної оборони. Про те, що небезпека нападу супротивника минула, і про порядок подальших дій розпорядження надійде по тих же каналах зв'язку, що і сигнал оповіщення.

При виникненні загрози нападу противника місцевими органами влади і штабами ЦЗ за допомогою засобів масової інформації передаються населенню постанови чи розпорядження про порядок дій. З цього часу радіо, телевізори повинні бути постійно включені для прийому нових повідомлень. У найкоротші терміни населення має вжити необхідних заходів захисту і включитися у виконання заходів, що проводяться ЦЗ [30].

ВИСНОВОК

У дипломній кваліфікаційній роботі магістра був розроблений засіб оброблення даних для аналізу результатів випробувань технічних об'єктів на базі дослідження моделювання безпеки автомобіля.

Мережа включає 25 робочих станцій. Розглянуто перспектива збільшення кількості вузлів.

В ході дипломного проєкту були вирішені наступні завдання:

- проведено порівняльний аналіз процесорів, технологій, інтерфейсів;
- виконано аналіз комп'ютерної архітектури, на підставі якого було розроблено технічне обґрунтування апаратної частини, була обрана архітектура;
- виконано обґрунтування програмних компонентів необхідних для розрахунків і виконана установка;
- виконано проєктування програми для зчитування даних з LS-Dyna;
- виконано створення кластера з використанням мови програмування Python та C++;
- виконано перевірку з використанням платформи Boinc .

Розроблений засіб дозволяє окремим користувачам легко і швидко взаємодіяти один з одним, а так само виконувати розрахунки для проєкту. Таке рішення дозволяє проводити всі необхідні розрахунки на базі робочих станцій на виробництві. І в цьому випадку зникає необхідність купувати обчислювальні потужності, що дозволяє істотно заощадити бюджет.

Розроблений засіб, за умови використання 25 робочих станцій, дозволяє проводити розрахунки на 65% швидше ніж на 1 робочій станції.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Будникова, Ю. А. Методика вычисления эффективных характеристик тканых материалов систем обеспечения пассивной безопасности легковых автомобилей [Текст]/ Ю. А. Будникова, Е. Д. Карташова, Е. А. Елескина // Модели, системы, сети в экономике, технике, природе и обществе. – 2013. – № 4 (8). – С. 158 –165.
2. Сухих, А.В., Исследование классификаций кластерных систем / А.В. Сухих, Н.С. Васяева // Кибернетика и программирование. – 2016. – № 2. – С. 20 - 27.
3. Высокая производительность вычислительного кластера [Электрон. ресурс]. – Режим доступа: <http://www.ncac.gwu.edu/hpcl/index.html>
4. Краш Тесты Электро- и Гибридных Автомобилей [Электрон. ресурс]. – Режим доступа: <https://hevcars.com.ua/crash-tests/>
5. Вычисления объединения Украина [Электрон. ресурс]. – Режим доступа: <https://secure.worldcommunitygrid.org/stat/viewGlobal.do>.
6. Обчислювальний комплекс СКІТ ІК НАН України [Електронний ресурс]. – Режим доступу: http://icybcluster.org.ua/index.php?lang_id=2&menu_id=5
7. Virtual Interface Architecture [Electronic resource]. – Access mode: https://www.researchgate.net/publication/3897314_Implementing_Virtual_Interface_Architecture_on_top_of_the_GM_message_passing_interface
8. Испытания компании NCCAC [Электрон. ресурс]. – Режим доступа: <http://www.ncac.gwu.edu/vml/archive/ncac/vehicle>
9. Описание вычисленной и программной платформы BOINC [Электрон. ресурс]. – Режим доступа: <http://www.boinc.ru/projects/projects.htm>
10. Рассмотрение LS-DYNA [Электрон. ресурс]. – Режим доступа: <http://www.ls-dyna.ru/>
11. Руководство по использованию LS-DYNA [Электрон. ресурс]. – Режим доступа: http://supercomputer.susu.ac.ru/users/instructions/instr_ls_dyna/

12. Цилькер Б.Я. Организация ЭВМ и систем : Учебник для вузов [Текст] / Б.Я. Цилькер, С.А. Орлов. – 2-е изд. – СПб.: Питер, 2018. – С. 688.
13. Воеводин, В.В. "Вычислительное дело и кластерные системы"/ В.В. Воеводин, С.А. Жуматий. – М.: Изд-во МГУ, 2017. – С. 150 с.
14. Эффективность кластера [Электрон. ресурс]. – Режим доступа: <http://cluster.linux-ekb.info/efficiency1.php>
15. Остапенко, В.В. Методичні вказівки по економічному обґрунтуванню дипломних проєктів для студентів спеціальностей 7.080403 «Програмне забезпечення автоматизованих систем», 7.080402 «Інформаційні технології проєктування» і 7.091501 «Комп'ютерні та інтелектуальні системи і мережі» всіх форм навчання [Текст] / В.В. Остапенко, Е.М. Касьян, Є.М. Сердюк. – Запоріжжя : ЗНТУ, 2009. – 22 с.
16. Правила улаштування електроустановок [Текст]. – Міненерговугілля України, 2017. – 617с.
17. ДСТУ 7237:2011 «Система стандартів безпеки праці. Електробезпека. Загальні вимоги та номенклатура видів захисту». – Київ: Держспоживстандарт України, 2011. – 10 с.
18. Жидецький В. Ц. Основи охорони праці [Текст] : підручник / В. Ц. Жидецький. – 5-те вид., доп. – К. : Знання, 2014. – 373 с.
19. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень. – Харків: Форт, 2000 – 24 с.
20. ДСН 3.3.6.037-99. Санитарные нормы производственного шума, ультразвука и инфразвука. – К. : МОЗ Украины, 1999. – 98 с.
21. Виробничий шум та вібрація: довідник спеціаліста служби охорони праці. – № 4, 2017. – С.34-54.
22. ДБН В.2.5-28:2018. Природне та штучне освітлення. Посібник: проблеми природного і штучного освітлення. – К: Етін, 2019. – 180 с.
23. ДБН В.2.5-67:2013. Отопление, вентиляция и кондиционирование. – Київ: Мінрегіон України, 2013. – 113 с.

24. Грибан В. Охорона праці / В. Грибан, А. Негодченко. – К. : Центр навчальної літератури, 2017. – 280 с.
25. Правила пожежної безпеки в Україні 2015. – Харків: Форт, 2015. – 124 с.
26. ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги». – Київ: Міністерство регіонального розвитку, будівництва та житлово-комунального господарства України, 2017. – 39 с.
27. НПАОП 40.1-1.32-01 «Правила будови електроустановок. Електрообладнання спеціальних установок». – Харків: Індустрія, 2010. – 118 с.
28. ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників. Загальні технічні вимоги». – Харків: Форт, 2004 – 48 с.
29. Стеблюк М. І. Цивільна оборона та цивільний захист [Текст] : навч. посіб. для вузів / М. І. Стеблюк. – К. : Знання, 2013, - 487 с.
30. Шоботов В. М. Цивільна оборона [Текст] : навчальний посібник / В. М. Шоботов. – Вид. 2-ге, перероб. – К. : Центр навчальної літератури, 2006. – 438 с.

ДОДАТОК А
Текст програми

Файл завантаження

```

from setuptools import setup, Extension
import sys
from sys import platform
useFemzip = True
debugging_mode = False
use_mingw=False
for ii in range(len(sys.argv)):
    if (sys.argv[ii] == "-c") and (sys.argv[ii+1] == "mingw32"):
        use_mingw=True
    if (sys.argv[ii] == "--compiler=mingw32"):
        use_mingw=True
srcs = ["src/python/wrapper.cpp"
        , "src/db/DB_Elements.cpp"
        , "src/db/DB_Nodes.cpp"
        , "src/db/DB_Parts.cpp"
        , "src/db/Element.cpp"
        , "src/db/Node.cpp"
        , "src/db/Part.cpp"
        , "src/dyna/DplotBuffer.cpp"
        , "src/dyna/dplot.cpp"
        , "src/utility/FileUtility.cpp"
        , "src/utility/IOUtility.cpp"
        , "src/utility/TextUtility.cpp"
        , "src/utility/MathUtility.cpp"]

include_dirs = []
compiler_args = []
if useFemzip:

    srcs.append("src/dyna/FemzipBuffer.cpp")

    if (platform == "win32"):
        lib_dirs = ['lib/Windows_VS2010_MT/x64']
        libs = ['femunziplib_standard_dyna','ipp_zlib','ippcoremt',
                'ippdcmt','ippsmt','ifwin','ifconsol','ippvmmmt','libmmt',
                'libirc','svml_dispmt','msvcrt']
        compiler_args.append("/DCD_USE_FEMZIP")
    else:
        lib_dirs = ['lib/Linux/64Bit']
        libs = ['femunzip_dyna_standard','ipp_z','ippcore',
                'ippdc','ipps','ifcore_pic','ifcoremt','imf',
                'ipgo','irc','svml','ippcore_l','stdc++','dl']
        compiler_args.append("-DCD_USE_FEMZIP")

```

else:

```
lib_dirs = []
libs = []
```

```
if (platform == "linux") or (platform == "linux2") or use_mingw:
    compiler_args.append("-std=c++11")
    compiler_args.append("-O3")
    if not use_mingw:
        compiler_args.append("-fPIC")
    if debugging_mode:
        compiler_args.append("-DCD_DEBUG")
else:
    if debugging_mode:
        compiler_args.append("/DCD_DEBUG")
```

Файл Dplot

```
#include <string>
#include "d3plot.h"
#include "D3plotBuffer.h"
#include "../db/Node.h"
#include "../db/Part.h"
#include "../db/Element.h"
#include "../db/DB_Nodes.h"
#include "../db/DB_Parts.h"
#include "../db/DB_Elements.h"
#include "../utility/TextUtility.h"
#include "../utility/IOUtility.h"
#include "../utility/FileUtility.h"

#ifdef CD_USE_FEMZIP
#include "FemzipBuffer.h"
#endif
```

```

Dplot::Dplot (string _filename,bool _useFemzip,vector<string>
_state_variables) {
    this->init_vars();
    const int bytesPerWord = 4;
    this->filename = _filename;
    this->useFemzip = _useFemzip;
    this->db_nodes = new DB_Nodes(this);
    this->db_parts = new DB_Parts();
    this->db_elements = new DB_Elements(this,this->db_nodes,this-
>db_parts);
    this->db_nodes->set_db_elements(this->db_elements);
#ifdef CD_USE_FEMZIP
    if(this->useFemzip){
        this->buffer = new FemzipBuffer(this->filename);
    } else {
        this->buffer = new D3plotBuffer(this->filename,bytesPerWord);
    }
#else
    if(this->useFemzip){
        throw("d3plot.cpp was compiled without femzip support.");
    }
    this->buffer = new D3plotBuffer(this->filename,bytesPerWord);
#endif

    this->buffer->read_geometryBuffer();
    this->read_header();
    this->read_geometry();
    this->read_states(_state_variables);
    I wouldn't do this, since we might need to read other state variables.
    if (this->buffer != NULL) {
        delete this->buffer;
        this->buffer = NULL;
    }
}

::~~Dplot(){

    if (this->buffer != NULL) {
        delete this->buffer;
        this->buffer = NULL;
    }
    if (this->db_nodes != NULL){
        delete this->db_nodes;
        this->db_nodes = NULL;
    }
}

```

```

if (this->db_parts != NULL) {
    delete this->db_parts;
    this->db_parts = NULL;
}
if (this->db_elements != NULL){
    delete this->db_elements;
    this->db_elements = NULL;
}

}

/*
 * Initialize the object vars.
 */
void Dplot::init_vars(){
    this->dyna_ndim = -1;
    this->dyna_numnp = -1;
    this->dyna_mdlopt = -1;
    this->dyna_mattyp = -1;
    this->dyna_nglbv = -1;
    this->dyna_nel2 = -1;
    this->dyna_nel4 = -1;
    this->dyna_nel48 = -1;
    this->dyna_nel8 = -1;
    this->dyna_nel20 = -1;
    this->dyna_nelth = -1;
    this->dyna_nmmtat = -1;
    this->dyna_nummat2 = -1;
    this->dyna_nummat4 = -1;
    this->dyna_nummat8 = -1;
    this->dyna_nummatth = -1;
    this->dyna_nv1d = -1;
    this->dyna_nv2d = -1;
    this->dyna_nv3d = -1;
    this->dyna_nv3dt = -1;
    this->dyna_maxint = -1;
    this->dyna_istrn = -1;
    this->dyna_neiph = -1;
    this->dyna_neips = -1;
    this->dyna_iu = -1;
    this->dyna_iv = -1;
    this->dyna_ia = -1;
    this->dyna_it = -1;
    this->dyna_idtdt = -1;
    this->dyna_narbs = -1;

```

```

this->dyna_ioshl1 = -1;
this->dyna_ioshl2 = -1;
this->dyna_ioshl3 = -1;
this->dyna_ioshl4 = -1;
this->dyna_extra = -1;
this->dyna_numprop = -1;
this->dyna_nmsph = -1;
this->dyna_ngpsph = -1;
this->dyna_ialemat = -1
this->nStates = -0;
vector<float> timesteps;
this->own_nel10 = false;
this->own_external_numbers_I8 = false;
this->wordPosition = 0;
this->wordsToRead = 0;
this->wordPositionStates = 0;
this->useFemzip=false;
this->femzip_state_offset=0;
this->plastic_strain_is_read = false;
this->plastic_strain_read = 0;
this->energy_is_read = false;
this->energy_read = 0;
this->strain_is_read = false;
this->strain_read = 0;
this->stress_is_read = false;
this->stress_read = 0;
this->disp_is_read = false;
this->disp_read = 0;
this->acc_is_read = false;
this->acc_read = 0;
this->vel_is_read = false;
this->vel_read = 0;}
void D3plot::read_header(){
#ifdef CD_DEBUG
cout << "> HEADER " << endl;
#endif
int filetype = this->buffer->read_int(11);
if(filetype > 1000){
    filetype -= 1000;
    own_external_numbers_I8 = true;
}
if(filetype != 1){
    throw("Wrong filetype "+to_string(this->buffer->read_int(11))+ " != 1 in
header of dplot");
}

```

```

    }

    this->dyna_title = this->buffer->read_str(0,10);
    int timestamp = this->buffer->read_int(10);
    time_t timestamp2 = (time_t) timestamp;
    this->dyna_datetime = asctime(localtime(&timestamp2));
    cout << timestamp << endl;
    cout << this->dyna_datetime << endl;
    this->dyna_ndim = this->buffer->read_int(15);
    this->dyna_mattyp = 0;
    if ( (this->dyna_ndim == 5) | (this->dyna_ndim == 7) ){
        this->dyna_mattyp = 1;
        this->dyna_ndim = 3;
    } else if (this->dyna_ndim == 4) {
        this->dyna_ndim = 3;
    } else if (this->dyna_ndim > 5) {
        throw("State data contains rigid road surface which can not be
handled.");
    } else {
        throw("Invalid parameter dyna_ndim="+to_string(this->dyna_ndim));
    }

    this->dyna_numnp = this->buffer->read_int(16);
    this->dyna_nglbv = this->buffer->read_int(18);

    this->dyna_iu = this->buffer->read_int(20);
    this->dyna_iv = this->buffer->read_int(21);
    this->dyna_ia = this->buffer->read_int(22);
    this->dyna_it = this->buffer->read_int(19);

    this->dyna_nel2 = this->buffer->read_int(28);
    this->dyna_nel4 = this->buffer->read_int(31);
    this->dyna_nel8 = this->buffer->read_int(23);
    this->dyna_nelth = this->buffer->read_int(40);
    this->dyna_nel48 = this->buffer->read_int(55);
    if (this->dyna_nel8 < 0){
        this->dyna_nel8 = abs(dyna_nel8);
        own_nel10 = true;
    }

    this->dyna_nmmtat = this->buffer->read_int(51);
    this->dyna_nummat2 = this->buffer->read_int(29);
    this->dyna_nummat4 = this->buffer->read_int(32);
    this->dyna_nummat8 = this->buffer->read_int(24);
    this->dyna_nummatth = this->buffer->read_int(41);

```

```

this->dyna_nv1d = this->buffer->read_int(30);
this->dyna_nv2d = this->buffer->read_int(33);
this->dyna_nv3d = this->buffer->read_int(27);
this->dyna_nv3dt = this->buffer->read_int(42);

this->dyna_neiph = this->buffer->read_int(34);
this->dyna_neips = this->buffer->read_int(35);
this->dyna_maxint = this->buffer->read_int(36);
if (this->dyna_maxint >= 0){
    this->dyna_mdlopt = 0;
} else if (this->dyna_maxint < 0){
    this->dyna_mdlopt = 1;
    this->dyna_maxint = abs(this->dyna_maxint);
}
if (this->dyna_maxint > 10000) {
    this->dyna_mdlopt = 2;
    this->dyna_maxint = abs(this->dyna_maxint)-10000;
}

this->dyna_narbs = this->buffer->read_int(39);

this->dyna_ioshl1 = this->buffer->read_int(43);
this->dyna_ioshl1 == 1000 ? this->dyna_ioshl1 = 1 : this->dyna_ioshl1 =
0;

this->dyna_ioshl2 = this->buffer->read_int(44);
this->dyna_ioshl2 == 1000 ? this->dyna_ioshl2 = 1 : this->dyna_ioshl2 =
0;

this->dyna_ioshl3 = this->buffer->read_int(45);
this->dyna_ioshl3 == 1000 ? this->dyna_ioshl3 = 1 : this->dyna_ioshl3 =
0;

this->dyna_ioshl4 = this->buffer->read_int(46);
this->dyna_ioshl4 == 1000 ? this->dyna_ioshl4 = 1 : this->dyna_ioshl4 =
0;

this->dyna_idtdt = this->buffer->read_int(56);
this->dyna_extra = this->buffer->read_int(57);
this->dyna_nmsph = this->buffer->read_int(37);
this->dyna_ngpsph = this->buffer->read_int(38);
this->dyna_ialemt = this->buffer->read_int(47);
if (this->dyna_extra > 0){
    this->dyna_nel20 = this->buffer->read_int(64);
} else {
    this->dyna_nel20 = 0;
}

```

```

if (this->dyna_idtdt > 100){
    this->dyna_istrn = this->dyna_idtdt % 10000;
} else {
    if (this->dyna_nv2d > 0) {
        if (this->dyna_nv2d - this->dyna_maxint *
            (6 * this->dyna_ioshl1 + this->dyna_ioshl2 + this->dyna_neips)
            + 8 * this->dyna_ioshl3 + 4 * this->dyna_ioshl4 > 1) {
            this->dyna_istrn = 1;
        } else {
            this->dyna_istrn = 0;
        }
    }
}
#ifdef CD_DEBUG
cout << "Title: " << this->dyna_title << endl;
cout << "nNodes : " << this->dyna_numnp << endl;
cout << "nElem2 : " << this->dyna_nel2 << endl;
cout << "nElem4 : " << this->dyna_nel4 << endl;
cout << "nElem8 : " << this->dyna_nel8 << endl;
cout << "nElem20: " << this->dyna_nel20 << endl;
cout << "nElemTh: " << this->dyna_nelth << endl;
cout << "nElem48: " << this->dyna_nel48 << endl;
cout << "nMats-Dyn: " << this->dyna_nmmt << endl;
cout << "nMats-Inp: " << this->dyna_nummat2
    +this->dyna_nummat4
    +this->dyna_nummat8
    +this->dyna_nummatth << endl;
cout << "nMat2 : " << this->dyna_nummat2 << endl;
cout << "nMat4 : " << this->dyna_nummat4 << endl;
cout << "nMat8 : " << this->dyna_nummat8 << endl;
cout << "nMatTh: " << this->dyna_nummatth << endl;
cout << "disp : " << this->dyna_iu << endl;
cout << "vel  : " << this->dyna_iv << endl;
cout << "accel: " << this->dyna_ia << endl;
cout << "temp : " << this->dyna_it << endl;
cout << "shell-stress: " << this->dyna_ioshl1 << endl;
cout << "shell-plstrn: " << this->dyna_ioshl2 << endl;
cout << "shell-forces: " << this->dyna_ioshl3 << endl;
cout << "shell-stuff : " << this->dyna_ioshl4 << endl;
cout << "shell-strn : " << this->dyna_istrn << endl;
cout << "nVar1D: " << this->dyna_nv1d << endl;
cout << "nVar2D: " << this->dyna_nv2d << endl;
cout << "nVar3D: " << this->dyna_nv3d << endl;
#endif
if(this->dyna_mattyp != 0) throw("MATTYPE != 0 can not be handled
(material type section).");

```

```

        if((this->dyna_nmsph != 0) | (this->dyna_ngpsph != 0)) throw("SPH mats
and elements can not be handled.");
        if(this->dyna_ialemat != 0) throw("ALE can not be handled.");
        if(this->dyna_nelth > 0) throw("Can not handle thick shell elements.");
        if(this->dyna_it != 0) throw("dyna_it != 0: Can not handle
temperatures.");
        if(own_external_numbers_I8) throw("Can not handle external ids with
double length.");
        if (this->dyna_nel48 > 0) throw("Can not handle 8 node shell elements.");
        if (this->dyna_nel20 > 0) throw("Can not handle 20 node solid
elements.");
        if (this->dyna_extra > 0){
            wordPosition = 64*2; // header has 128 words
        } else {
            wordPosition = 64; // header has 64 words
        } }
void Dplot::read_geometry(){

#ifdef CD_DEBUG
cout << "> GEOMETRY" << endl;
#endif
vector< vector <float> > buffer_nodes = this->read_geometry_nodes();
vector< vector <int> > buffer_elems8 = this->read_geometry_elem8();
if(dyna_nelth > 0) wordPosition += 9*dyna_nelth;
vector< vector <int> > buffer_elems2 = this->read_geometry_elem2();
vector< vector <int> > buffer_elems4 = this->read_geometry_elem4();
vector< vector<int> > buffer_numbering = this-
>read_geometry_numbering();
    if(!isFileEnding(wordPosition)){
        //cout << this->buffer->read_float(wordPosition) << endl;
        throw("Anticipated file ending wrong in geometry section.");
    }
    wordPosition++;
    this->buffer->free_geometryBuffer();
    this->buffer->read_partBuffer();
    if(this->useFemzip)
        wordPosition = 1;
    this->read_geometry_parts();
    if(!isFileEnding(wordPosition)){
        //cout << this->buffer->read_float(wordPosition) << endl;
        throw("Anticipated file ending wrong in geometry section.");
    }
    this->buffer->free_partBuffer();
#ifdef CD_DEBUG
cout << "Adding nodes ... ";

```

```

#endif
if(buffer_numbering[0].size() != buffer_nodes.size())
    throw("Buffer node-numbering and buffer-nodes have different sizes.");
for(unsigned int ii=0; ii < buffer_nodes.size() ;ii++){

    this->db_nodes->add_node(buffer_numbering[0][ii],buffer_nodes[ii]);
}
#ifdef CD_DEBUG
cout << db_nodes->size() << " done." << endl;
#endif
#ifdef CD_DEBUG
cout << "Adding beams ... ";
#endif
for(unsigned int ii=0; ii < buffer_elems2.size() ;ii++){
    this->db_elements-
>add_element(BEAM,buffer_numbering[2][ii],buffer_elems2[ii]);
}
#ifdef CD_DEBUG
cout << db_elements->size() << " done." << endl;
#endif
// Shells
#ifdef CD_DEBUG
cout << "Adding shells ... ";
#endif
for(unsigned int ii=0; ii < buffer_elems4.size() ;ii++){
    this->db_elements-
>add_element(SHELL,buffer_numbering[3][ii],buffer_elems4[ii]);

}
#ifdef CD_DEBUG
cout << db_elements->size() << " done." << endl;
#endif

// Solids
#ifdef CD_DEBUG
cout << "Adding solids ... ";
#endif
for(unsigned int ii=0; ii < buffer_elems8.size() ;ii++){

    this->db_elements-
>add_element(SOLID,buffer_numbering[1][ii],buffer_elems8[ii]);

}
#ifdef CD_DEBUG
cout << db_elements->size() << " done." << endl;

```

```

#endif }
vector<vector<float>> D3plot::read_geometry_nodes(){
#ifdef CD_DEBUG
cout << "Reading nodes ... ";
#endif
wordsToRead = dyna_numnp*dyna_ndim;
vector<vector<float>> buffer_nodes(dyna_numnp);
int jj = 0;
for (int ii=wordPosition;ii<wordPosition+wordsToRead;ii+=3){
    vector<float> coords(3);
    coords[0] = buffer->read_float(ii);
    coords[1] = buffer->read_float(ii+1);
    coords[2] = buffer->read_float(ii+2);
    buffer_nodes[jj] = coords;
    jj++;
}
wordPosition += wordsToRead;
#ifdef CD_DEBUG
cout << "done." << endl;
#endif
return buffer_nodes; }
vector<vector<int>> Dplot::read_geometry_elem8(){
if(dyna_nel8 == 0) return vector<vector<int>>();
#ifdef CD_DEBUG
cout << "Reading elems8 ... ";
#endif
const int nVarsElem8 = 9;
vector<vector<int>> buffer_elems8(dyna_nel8);
wordsToRead = nVarsElem8*dyna_nel8;
int iElement = 0;
int iData = 0;
for
(ii=wordPosition;ii<wordPosition+wordsToRead;ii+=nVarsElem8){
    iData = 0;
    vector<int> elemData(nVarsElem8);
    for(int jj=ii;jj<ii+nVarsElem8;jj++){
        elemData[iData] = buffer->read_int(jj);
        iData++;
    }
    buffer_elems8[iElement] = elemData;
    iElement++;
}
wordPosition += wordsToRead;
if(own_nel10) wordPosition += 2*dyna_nel8;

```

```

#ifdef CD_DEBUG
cout << "done." << endl;
#endif
return buffer_elems8;
}
vector<vector<int>> D3plot::read_geometry_elem4(){
    if(dyna_nel4 == 0) return vector<vector<int>>();
#ifdef CD_DEBUG
cout << "Reading elems4 ... ";
#endif
const int nVarsElem4 = 5;
vector<vector<int>> buffer_elems4(dyna_nel4);
wordsToRead = nVarsElem4*dyna_nel4;
int iElement = 0;
int iData = 0;
    for
(ii=wordPosition;ii<wordPosition+wordsToRead;ii+=nVarsElem4){
        iData = 0;
        vector<int> elemData(nVarsElem4);
        for(int jj=ii;jj<ii+nVarsElem4;jj++){
            elemData[iData] = buffer->read_int(jj);
            iData++;
        }
        buffer_elems4[iElement] = elemData;
        iElement++;
    }

    // Update word position
    wordPosition += wordsToRead;

#ifdef CD_DEBUG
cout << "done." << endl;
#endif

    return buffer_elems4;
}
vector<vector<int>> Dplot::read_geometry_elem2(){
    if(dyna_nel2 == 0) return vector<vector<int>>();
#ifdef CD_DEBUG
cout << "Reading elems2 ... ";
#endif
const int nVarsElem2 = 6;
vector<vector<int>> buffer_elems2(dyna_nel2);
wordsToRead = nVarsElem2*dyna_nel2;
int iElement = 0;

```

```

        for
        ii=wordPosition;ii<wordPosition+wordsToRead;ii+=nVarsElem2){
            iData = 0;
            vector<int> elemData(nVarsElem2);
            for(int jj=ii;jj<ii+nVarsElem2;jj++){
                elemData[iData] = buffer->read_int(jj);
                iData++;
            }
            vector<int> elemData(3);
            elemData[0] = buffer->read_int(ii);
            elemData[1] = buffer->read_int(ii+1);
            elemData[2] = buffer->read_int(ii+5);
            buffer_elems2[iElement] = elemData;
            iElement++;
        }
        wordPosition += wordsToRead;
#ifdef CD_DEBUG
        cout << "done." << endl;
#endif

        return buffer_elems2;
    }

vector<vector<int>>> D3plot::read_geometry_numbering(){

    int nHeaderWords = 10;
    int kk = 10+dyna_numnp+dyna_nel2+dyna_nel4+dyna_nel8+dyna_nelth;
    if(dyna_narbs == kk){
        nHeaderWords = 10;
    }
    else if(dyna_narbs ==
kk+dyna_nummat2+dyna_nummat4+dyna_nummat8+dyna_nummatth ){
        nHeaderWords = 16;
    } else {
        throw("Inconsistent definition of dyna_narbs detected.");
    }
    if(dyna_narbs == 0) return vector<vector<int>>>();
#ifdef CD_DEBUG
    cout << "Reading numbering ... ";
#endif
    int nsort = buffer->read_int(wordPosition);
    int nsrh = buffer->read_int(wordPosition+1);
    if(nsrh != dyna_numnp+abs(nsort)) throw("nsrh != nsort + numnp is
inconsistent in dyna file.");
    // pointer to elem2 numbering

```

```

    int nsrb = buffer->read_int(wordPosition+2);
    if(nsrb != nsrh+dyna_nel8) throw("nsrb != nsrh + nel8 is inconsistent in
dyna file.");
    int nsrs = buffer->read_int(wordPosition+3);
    if(nsrs != nsrb+dyna_nel2) throw("nsrs != nsrb + nel2 is inconsistent in
dyna file.");
    int nsrt = buffer->read_int(wordPosition+4);
    if(nsrt != nsrs+dyna_nel4) throw("nsrt != nsrs + nel4 is inconsistent in
dyna file.");
    if(buffer->read_int(wordPosition+5) != dyna_numnp) throw("Number of
nodes is not defined consistent in d3plot geometry section.");

```

```

    int                                     nMaterials                                     =
dyna_nummat2+dyna_nummat4+dyna_nummat8+dyna_nummatth;
    vector<vector<int>> idvector(4);
    if (nsort < 0) {
        wordPosition += 16;
    } else {
        wordPosition += 10;
    }
    //wordPosition += 16; // header length is 16
    wordsToRead = dyna_numnp;
    vector<int> nodeIDs(wordsToRead);
    int jj = 0;
    for(int ii=wordPosition;ii<wordPosition+wordsToRead;ii++){
        nodeIDs[jj] = buffer->read_int(ii);
        jj++;
    }
    idvector[0] = nodeIDs;

    // Solid IDs
    wordPosition += wordsToRead;
    wordsToRead = dyna_nel8;
    vector<int> solidIDs(wordsToRead);
    jj = 0;
    for(int ii=wordPosition;ii<wordPosition+wordsToRead;ii++){
        solidIDs[jj] = buffer->read_int(ii);
        jj++;
    }
    idvector[1] = solidIDs;
    wordPosition += wordsToRead;
    wordsToRead = dyna_nel2;
    vector<int> beamIDs(wordsToRead);
    jj = 0;
    for(int ii=wordPosition;ii<wordPosition+wordsToRead;ii++){

```

```

    beamIDs[jj] = buffer->read_int(ii);
    jj++;
}
idvector[2] = beamIDs;
wordPosition += wordsToRead;
wordsToRead = dyna_nel4;
vector<int> shellIDs(wordsToRead);
jj = 0;
for(int ii=wordPosition;ii<wordPosition+wordsToRead;ii++){
    shellIDs[jj] = buffer->read_int(ii);
    jj++;
}
idvector[3] = shellIDs;
wordPosition += wordsToRead;
wordPosition += dyna_nelth;
// not read ...
vector<int> internalPartIDs(nMaterials);
vector<int> externalPartIDs(nMaterials);
wordsToRead = 3*dyna_nmmat;
jj = 0;
for(int
ii=wordPosition+dyna_nmmat;ii<wordPosition+dyna_nmmat+nMaterials;ii++){
    externalPartIDs[jj] = buffer->read_int(ii);
}

jj = 0;
for(int
ii=wordPosition+2*dyna_nmmat;ii<wordPosition+2*dyna_nmmat+nMaterials;ii+
+){
    internalPartIDs[jj] = buffer->read_int(ii);
}

// update position
wordPosition += wordsToRead;
/*
 * the whole numbering section should have length narbs
 * but the procedure here should do the same ... hopefully
 */
//wordPosition += dyna_narbs;

// extra node elements
// 10 node solids: 2 node conn
if(own_nel10)
    wordPosition += 2*abs(dyna_nel8);
// 8 node shells: 4 node conn

```

```

    if(dyna_nel48 > 0)
        wordPosition += 5*dyna_nel48;
    // 20 node solids: 12 node conn
    if((dyna_extra > 0) & (dyna_nel20 > 0))
        wordPosition += 13*dyna_nel20;
#ifdef CD_DEBUG
    cout << "done." << endl;
#endif

    return idvector;

}

/*
 * Read the part names from the geometry section.
 */
void D3plot::read_geometry_parts(){

#ifdef CD_DEBUG
    cout << "Reading parts ... " ;
#endif

    int ntype = this->buffer->read_int(wordPosition);
    if (ntype != 90001){
        throw("ntype must be 90001 in part section.");
    }

    this->dyna_numprop = this->buffer->read_int(wordPosition+1);
    if (this->dyna_numprop < 0)
        throw("negative number of parts in part section makes no sense.");
    for(int ii=0; ii<this->dyna_numprop; ii++){

        int start = (wordPosition + 1) + ii*19 + 1;
        int partID = this->buffer->read_int(start);
        string partName = this->buffer->read_str(start+1,18);

        this->db_parts->add_part(this->db_parts->size()+1,partID)-
>set_name(partName);

    }

#ifdef CD_DEBUG
    cout << this->db_parts->size() << " done." << endl;
#endif
}

```

```

// update position
wordPosition += 1 + (this->dyna_numprop+1)*19+1;

}

/*
 * Check for the file ending marker. That is a float
 * of -999999.
 */
bool D3plot::isFileEnding(int iWord){
    if (this->buffer->read_float(iWord)+999999.0f == 0.)
        return true;
    return false;
}

/*
 * Checks if the states are alle fine and sets a few variables.
 *
 */
void D3plot::read_states_init(){

#ifdef CD_DEBUG
    cout << "> STATES INIT" << endl;
#endif

    wordPositionStates = wordPosition;

    int iState = 0;
    int nVarsNodes = dyna_ndim*(dyna_iu+dyna_iv+dyna_ia)*dyna_numnp;
    int nVarsElems =
    dyna_nel2*dyna_nv1d+dyna_nel4*dyna_nv2d+dyna_nel8*dyna_nv3d;

    int nDeletionVars = -1;
    if (dyna_mdlopt == 0){
        // ok
    } else if(dyna_mdlopt == 1){
        nDeletionVars = dyna_numnp;
    } else if(dyna_mdlopt == 2){
        nDeletionVars = dyna_nel2+dyna_nel4+dyna_nel8+dyna_nelth;
    } else {
        throw("Parameter   mdlopt:"+to_string(dyna_mdlopt)+"   makes   no
sense.");
    }
}

```

```

// Loop through state files
this->buffer->init_nextState();
bool firstFileDone = false;
while(this->buffer->has_nextState()){

    this->buffer->read_nextState();

    /*
    if(firstFileDone){
        wordPosition = 0;
    }*/
    // No femzip case
    if(!(this->useFemzip) & firstFileDone){
        wordPosition = 0;
    }
    // femzip case
    if(this->useFemzip){
        // 0 = endmark
        // 1 = ntype = 90001
        // 2 = numprop
        int dyna_numprop_states = this->buffer->read_int(2);
        if(this->dyna_numprop != dyna_numprop_states)
            throw("Numprop in geometry section != numprop in states section!");
        wordPosition = 1;
        wordPosition += 1 + (this->dyna_numprop+1)*19+1;
    }

    // Loop through states
    while(!this->isFileEnding(wordPosition)){

        float state_time = buffer->read_float(wordPosition);
        this->timesteps.push_back(state_time);

        // update position
        wordPosition += nVarsNodes + nVarsElems + nDeletionVars +
dyna_nglbv + 1;

        iState++;
    }

    firstFileDone = true;
}

```

```

this->buffer->end_nextState();
#ifdef CD_DEBUG
cout << "nTimeSteps: " << this->timesteps.size() << endl;
#endif
}

/*
 * Parse the user input for state reading.
 */
void D3plot::read_states_parse(vector<string> _variables){

    // Safety reset
    this->disp_read = 0;
    this->vel_read = 0;
    this->acc_read = 0;
    this->stress_read = 0;
    this->strain_read = 0;
    this->energy_read = 0;
    this->plastic_strain_read = 0;

    this->history_shell_read.clear();
    this->history_shell_mode.clear();

    this->history_solid_read.clear();
    this->history_solid_mode.clear();

    for(unsigned int ii=0; ii<_variables.size(); ++ii){
        // Displacement
        if(_variables[ii].find("disp") != string::npos){
            if( dyna_iu == 0)
                throw("Unable to read displacements, since there are none.");
            this->disp_read = read_states_parse_readMode(_variables[ii]);

            if(this->disp_is_read){
#ifdef CD_DEBUG
                cout << "disp already loaded." << endl;
#endif
                this->disp_read = 0;
            }
        }
        // Velocity
    } else if(_variables[ii].find("vel") != string::npos){
        if( dyna_iv == 0)
            throw("Unable to read velocities, since there are none.");
    }
}

```

```

this->vel_read = read_states_parse_readMode(_variables[ii]);

if(this->vel_is_read){
    #ifdef CD_DEBUG
        cout << "vel already loaded." << endl;
    #endif
    this->vel_read = 0;
}
// Acceleration
} else if(_variables[ii].find("accel") != string::npos){
    if( dyna_ia == 0)
        throw("Unable to read accelerations, since there are none.");
    this->acc_read = read_states_parse_readMode(_variables[ii]);

    if(this->acc_is_read){
        #ifdef CD_DEBUG
            cout << "accel already loaded." << endl;
        #endif
        this->acc_read = 0;
    }
// Stress
} else if(_variables[ii].find("stress") != string::npos){
    if( dyna_ioshl1 == 0)
        cout << "Warning: There are no shell-stresses in the file." << endl;
    this->stress_read = read_states_parse_readMode(_variables[ii]);

    if(this->stress_is_read){
        #ifdef CD_DEBUG
            cout << "stress already loaded." << endl;
        #endif
        this->stress_read = 0;
    }
// Plastic Strain
// must be before strain !!!!
} else if(_variables[ii].find("plastic_strain") != string::npos){
    if( dyna_ioshl2 == 0)
        cout << "Warning: There are no shell-plastic-strains in the file." <<
endl;
    this->plastic_strain_read = read_states_parse_readMode(_variables[ii]);

    if(this->plastic_strain_is_read){
        #ifdef CD_DEBUG
            cout << "plastic strain already loaded." << endl;
        #endif
        this->plastic_strain_read = 0;
    }

```

```

    }
    // Strain
} else if(_variables[ii].find("strain") != string::npos){
    if( dyna_istrn == 0)
        throw("Unable to read strains, since there are none.");
    this->strain_read = read_states_parse_readMode(_variables[ii]);

    if(this->strain_is_read){
        #ifdef CD_DEBUG
        cout << "strain already loaded." << endl;
        #endif
        this->strain_read = 0;
    }
}
// Internal Energy
} else if(_variables[ii].find("energy") != string::npos){
    if( dyna_ioshl4 == 0)
        throw("Unable to read energies, since there are none.");
    this->energy_read = read_states_parse_readMode(_variables[ii]);

    if(this->energy_is_read){
        #ifdef CD_DEBUG
        cout << "energy already loaded." << endl;
        #endif
        this->energy_read = 0;
    }
}
// History variables
} else if(_variables[ii].find("history") != string::npos){

    // retrieve history var indexes
    vector<unsigned int> hist_vars =
TextUtility::extract_integers(_variables[ii]);
    if(hist_vars.size() < 1)
        throw("No history variable index specified. Please input at least one
number seperated by spaces.");
    unsigned int var_mode = read_states_parse_readMode(_variables[ii]);

    /* SHELLS */
    if(_variables[ii].find("shell") != string::npos){

        // Check: already loaded
        for(size_t jj=0; jj<this->history_shell_is_read.size(); ++jj){
            vector<unsigned int>::iterator kk = find(hist_vars.begin(),
hist_vars.end(), this->history_shell_is_read[jj]);
            if ( kk != hist_vars.end() ){
                #ifdef CD_DEBUG

```

```

        cout << "history variable " << *kk << " already loaded for shells."
<< endl;
        #endif
        hist_vars.erase(kk);
    }
}

// Check: already defined in an argument previously
for(size_t jj=0; jj<this->history_shell_read.size(); ++jj){
    vector<unsigned int>::iterator kk = find(hist_vars.begin(),
hist_vars.end(), this->history_shell_read[jj]);
    if ( kk != hist_vars.end() ){
        cout << "Warning: trying to read history variable " << *kk << "
twice for shells, using only first occurrence." << endl;
        hist_vars.erase(kk);
    }
}

// Check: var index beyond limit neips
for(size_t jj=0; jj < hist_vars.size(); ++jj){

    if(hist_vars[jj] < 1){
        throw("History variable index must be at least 1.");
    }
    if( hist_vars[jj] > this->dyna_neips ){
        cout << "Warning: history variable " << hist_vars[jj] << " exceeds
the limit for shells of " << this->dyna_neips << endl;
    }
}

// Save var indexes and var modes
for(size_t jj=0; jj<hist_vars.size(); ++jj){
    this->history_shell_read.push_back(hist_vars[jj]);
    this->history_shell_mode.push_back(var_mode);
}

/* SOLIDS */
} else if(_variables[ii].find("solid") != string::npos) {

    // Check: already loaded
    for(size_t jj=0; jj<this->history_solid_is_read.size(); ++jj){
        vector<unsigned int>::iterator kk = find(hist_vars.begin(),
hist_vars.end(), this->history_solid_is_read[jj]);
        if ( kk != hist_vars.end() ){
            #ifdef CD_DEBUG

```

```

        cout << "history variable " << *kk << " already loaded for solids."
<< endl;
        #endif
        hist_vars.erase(kk);
    }
}

// Check: already defined in an argument previously
for(size_t jj=0; jj<this->history_solid_read.size(); ++jj){
    vector<unsigned int>::iterator kk = find(hist_vars.begin(),
hist_vars.end(), this->history_solid_read[jj]);
    if ( kk != hist_vars.end() ){
        cout << "Warning: trying to read history variable " << *kk << "
twice for solids, using only first occurrence." << endl;
        hist_vars.erase(kk);
    }
}

// Check: var index beyond limit neiph
for(size_t jj=0; jj < hist_vars.size(); ++jj){

    if(hist_vars[jj] < 1){
        throw("History variable index must be at least 1.");
    }
    if( hist_vars[jj] > this->dyna_neiph ){
        cout << "Warning: history variable " << hist_vars[jj] << " exceeds
the limit for solids of " << this->dyna_neiph << endl;
    }
}

// save var indexes
for(size_t jj=0; jj<hist_vars.size(); ++jj){
    this->history_solid_read.push_back(hist_vars[jj]);
    this->history_solid_mode.push_back(var_mode);
}

// unknown element type
} else {
    throw("Please specify the element type for all history variables as
shell or solid");
}

} else {
    throw("Reading of variable:"+_variables[ii]+" is undefined");
} // if:variable.find

```

```

    } //for:variables

}

/*
 * Returns the int code for the read mode of the state variables in the
d3plot.
 * Modes are: min,max,outer,mid,innter,mean
 */
unsigned int D3plot::read_states_parse_readMode(string _variable){

    if(_variable.find("max") != string::npos){
        return 1;
    } else if(_variable.find("min") != string::npos){
        return 2;
    } else if(_variable.find("outer") != string::npos){
        return 3;
    } else if(_variable.find("mid") != string::npos){
        return 4;
    } else if(_variable.find("inner") != string::npos){
        return 5;
    } else if(_variable.find("mean") != string::npos){
        return 6;
    } else {
        return 6; // std is mean
    }

}

void Dplot::read_states(vector<string> _variables){

#ifdef CD_DEBUG
    cout << "> STATES" << endl;
    for(unsigned int ii=0; ii<_variables.size(); ++ii)
        cout << "variable: " << _variables[ii] << endl;
#endif

    if( (_variables.size() < 1) & (this->timesteps.size() > 0))
        throw("The list of state variables to load is empty.");
    read_states_parse(_variables);

    // Check if variables are already read.
    // If just 1 is not read yet, the routine runs through.
    // Will not work first time :P since we must run through at least once

```

```

if( (this->disp_read
    +this->vel_read
    +this->acc_read
    +this->plastic_strain_read
    +this->energy_read
    +this->strain_read
    +this->stress_read
    +this->history_shell_read.size()
    +this->history_solid_read.size() == 0) & (this->timesteps.size() != 0))
    return;

// Calculate loop properties
unsigned int iState = 0;
int nVarsNodes = dyna_ndim*(dyna_iu+dyna_iv+dyna_ia)*dyna_numnp;
int nVarsElems =
dyna_nel2*dyna_nv1d+dyna_nel4*dyna_nv2d+dyna_nel8*dyna_nv3d;

// Variable Deletion table
int nDeletionVars = -1;
if (dyna_mdlopt == 0){
    // ok
} else if(dyna_mdlopt == 1){
    nDeletionVars = dyna_numnp;
} else if(dyna_mdlopt == 2){
    nDeletionVars = dyna_nel2+dyna_nel4+dyna_nel8+dyna_nelth;
} else {
    throw("Parameter   mdlopt:"+to_string(dyna_mdlopt)+"   makes   no
sense.");
}

// Checks for timesteps
bool timesteps_read = false;
if(this->timesteps.size() < 1)
    timesteps_read = true;

bool firstFileDone = false;

// Check for first time
// Makes no difference for D3plotBuffer but for
// the FemzipBuffer.
if(this->timesteps.size() < 1){
    this->buffer->init_nextState();
    wordPositionStates = this->wordPosition;
} else{
    this->buffer->rewind_nextState();

```

```

    this->wordPosition = wordPositionStates;
}

// Loop over state files
while(this->buffer->has_nextState()){

    this->buffer->read_nextState();

    // Not femzip case
    if((!this->useFemzip) & firstFileDone){
        wordPosition = 0;
    }
    // femzip case
    if(this->useFemzip){
        // 0 = endmark
        // 1 = ntype = 90001
        // 2 = numprop
        int dyna_numprop_states = this->buffer->read_int(2);
        if(this->dyna_numprop != dyna_numprop_states)
            throw("Numprop in geometry section != numprop in states section!");
        wordPosition = 1; // endl symbol at 0 in case of femzip ...
        wordPosition += 1 + (this->dyna_numprop+1)*19+1;
        this->femzip_state_offset = wordPosition;
    }

    // Loop through states
    while(!this->isFileEnding(wordPosition)){

        if(timesteps_read){
            float state_time = buffer->read_float(wordPosition);
            this->timesteps.push_back(state_time);
            #ifdef CD_DEBUG
                printf("State %u: %.4f\n",iState,state_time);
                //cout << "State " << iState << ": " << state_time << endl;
            #endif
        }

        // NODE - DISP
        if(dyna_iu & (this->disp_read != 0) ){
            read_states_displacement();
        }

        // NODE - VEL
        if(dyna_iv & (this->vel_read != 0) ){
            read_states_velocity();
        }
    }
}

```

```

    }

    // NODE - ACCEL
    if(dyna_ia & (this->acc_read != 0) ){
        read_states_acceleration();
    }

    // ELEMENT - STRESS, STRAIN, ENERGY, PLASTIC STRAIN
    if( (this->stress_read != 0)
        |(this->strain_read != 0)
        |(this->energy_read != 0)
        |(this->plastic_strain_read != 0)
        |(this->history_shell_read.size() != 0)
        |(this->history_solid_read.size() != 0)){

        // Element 4
        read_states_elem4(iState);
        // Element 8
        read_states_elem8(iState);

    }

    // update position
    wordPosition += nVarsNodes + nVarsElems + nDeletionVars +
dyna_nglbv + 1;

    iState++;
}

firstFileDone = true;
}

this->buffer->end_nextState();

// Set, which variables were read
if( this->disp_read != 0 ){
    this->disp_is_read = true;
}
if( this->vel_read != 0 ){
    this->vel_is_read = true;
}
if( this->plastic_strain_read != 0 ){
    this->plastic_strain_is_read = true;
}
if( this->energy_read != 0 ){

```

```

    this->energy_is_read = true;
}
if( this->strain_read != 0 ){
    this->strain_is_read = true;
}
if( this->stress_read != 0 ){
    this->stress_is_read = true;
}
for(size_t ii=0; ii<this->history_shell_read.size(); ++ii){
    this->history_shell_is_read.push_back(this->history_shell_read[ii]);
}
for(size_t ii=0; ii<this->history_solid_read.size(); ++ii){
    this->history_solid_is_read.push_back(this->history_solid_read[ii]);
}
}

void Dplot::read_states_displacement(){

    if(dyna_iu != 1)
        return;

    int start = wordPosition + dyna_nglbv + 1;
    wordsToRead = dyna_numnp*dyna_ndim;
    int iNode = 1;

    for(int ii = start;ii < start+wordsToRead; ii+=dyna_ndim){

        Node* node = this->db_nodes->get_nodeByIndex(iNode);
        vector<float> _disp;

        for(int jj=ii;jj<ii+dyna_ndim;jj++)
            _disp.push_back(this->buffer->read_float(jj));

        node->add_disp(_disp);

        iNode++;
    }
}

/*
 * Read the node velocity.
 *
 */

```

```

void D3plot::read_states_velocity(){

    if(dyna_iv != 1)
        return;

    int start = 1 + dyna_nglbv + (dyna_iu) * dyna_numnp * dyna_ndim +
this->femzip_state_offset;
    wordsToRead = dyna_numnp*dyna_ndim;
    int iNode = 1;

    for(int ii = start;ii < start+wordsToRead; ii+=dyna_ndim){

        Node* node = this->db_nodes->get_nodeByIndex(iNode);
        vector<float> _vel;

        for(int jj=ii; jj<ii+dyna_ndim; jj++)
            _vel.push_back(this->buffer->read_float(jj));

        node->add_vel(_vel);

        iNode++;
    }

}

void Dplot::read_states_acceleration(){

    if(dyna_ia != 1)
        return;

    int start = 1 + dyna_nglbv + (dyna_iu+dyna_iv) * dyna_numnp *
dyna_ndim + this->femzip_state_offset;
    wordsToRead = dyna_numnp*dyna_ndim;
    int iNode = 1;

    for(int ii = start;ii < start+wordsToRead; ii+=dyna_ndim){

        Node* node = this->db_nodes->get_nodeByIndex(iNode);
        vector<float> _accel;

        for(int jj=ii; jj<ii+dyna_ndim; jj++)
            _accel.push_back(this->buffer->read_float(jj));

        node->add_accel(_accel);
    }

}

```

```

        iNode++;
    }

}

void Dplot::read_states_elem8(unsigned int iState){

    if((dyna_nv3d <= 0) & (dyna_nel8 <= 0))
        return;

    int start = 1 + dyna_nglbv + (dyna_iu+dyna_iv+dyna_ia) * dyna_numnp *
dyna_ndim + this->femzip_state_offset;
    wordsToRead = dyna_nv3d * dyna_nel8;

    int iElement = 1;
    int delta = 7+dyna_neiph;
    for(int ii = start;ii < start+wordsToRead; ii+=delta){

        Element*          element          =          this->db_elements-
>get_elementByIndex(SOLID,iElement);

        // stress tensor
        if(this->stress_read){
            vector<float> sigma(6);
            sigma[0] = this->buffer->read_float(ii);
            sigma[1] = this->buffer->read_float(ii+1);
            sigma[2] = this->buffer->read_float(ii+2);
            sigma[3] = this->buffer->read_float(ii+3);
            sigma[4] = this->buffer->read_float(ii+4);
            sigma[5] = this->buffer->read_float(ii+5);
            element->add_stress(sigma);
        }

        // plastic strain
        if(this->plastic_strain_read){
            element->add_plastic_strain(this->buffer->read_float(ii+6));
        }

        // strain tensor
        if( (dyna_istrn == 1) & this->strain_read ){
            vector<float> strain(6);
            strain[0] = this->buffer->read_float(ii+delta-6);
            strain[1] = this->buffer->read_float(ii+delta-5);

```

```

        strain[2] = this->buffer->read_float(ii+delta-4);
        strain[3] = this->buffer->read_float(ii+delta-3);
        strain[4] = this->buffer->read_float(ii+delta-2);
        strain[5] = this->buffer->read_float(ii+delta-1);
        element->add_strain(strain);
    }

    // no energy ...

    // history variables
    if(history_solid_read.size()){
        vector<float> history_vars;
        history_vars.assign(this->history_solid_read.size(),0.f);

        for(size_t jj=0; jj<this->history_solid_read.size(); ++jj){

            // Skip if over limit
            if(this->history_solid_read[jj] > this->dyna_neiph)
                continue;

            history_vars[jj] = this->buffer->
read_float(ii+6+history_solid_read[jj]);

            } // loop:history
            element->add_history_vars(history_vars,iState);
        } // if:history

        iElement++;
    }

}

/*
 * Read the data of the 4 node solids.
 * > Strain Tensor
 * > Strain Mises
 * > Stress Tensor
 * > Stress Mises
 * > Eq. Plastic Strain (done)
 *
 */
void Dplot::read_states_elem4(unsigned int iState){

    if((dyna_istrn != 1) & (dyna_nv2d <= 0) & (dyna_nel4>0))

```

```

return;

// prepare looping
int start = 1 + dyna_nglbv
    + (dyna_iu+dyna_iv+dyna_ia) * dyna_numnp * dyna_ndim
    + dyna_nv3d * dyna_nel8
    + dyna_nv1d * dyna_nel2
    + this->femzip_state_offset;

wordsToRead = dyna_nv2d*dyna_nel4;

int iElement = 1;
for(int ii = start; ii < start+wordsToRead; ii+=dyna_nv2d){

    // preallocate layer vars
    vector<float> stress(6);
    float plastic_strain = 0;
    vector<float> history_vars;
    if(this->history_shell_read.size()){
        history_vars.assign(this->history_shell_read.size(),0.f);
    }

    // Loop: layers
    for(int iLayer = 0; iLayer < dyna_maxint; iLayer++){

        int layerStart = ii + iLayer*(7 + dyna_neips);

        // layers: plastic strain (in/out,min/mid/max)
        if( (this->plastic_strain_read) & (dyna_ioshl2)){
            float _tmp = this->buffer->read_float(layerStart+6);

            // max
            if(this->plastic_strain_read == 1)
                plastic_strain = (plastic_strain>_tmp) ? plastic_strain : _tmp;
            // min
            else if(this->plastic_strain_read == 2)
                plastic_strain = (plastic_strain<_tmp) ? plastic_strain : _tmp;
            // out
            else if((this->plastic_strain_read == 3) & (iLayer == dyna_maxint -1)
)
                plastic_strain = _tmp;
            // mid
            else if((this->plastic_strain_read == 4) & (iLayer == dyna_maxint / 2)
)
                plastic_strain = _tmp;

```

```

// in
else if((this->plastic_strain_read == 5) & (iLayer == 1) )
    plastic_strain = _tmp;
// mean
else if((this->plastic_strain_read == 6))
    plastic_strain = _tmp;
else
    throw("Unknown    var_mode    plastic    strain:"+to_string(this-
>plastic_strain_read));

    continue;
}
if( (this->stress_read) & (dyna_ioshl1)){
// max
if(this->stress_read == 1) {
    float _tmp;
    for(int jj=0; jj<6; jj++){
        _tmp = this->buffer->read_float(layerStart+jj);
        if(iLayer == 0){
            stress[jj] = _tmp;
        } else {
            stress[jj] = (_tmp > stress[jj]) ? _tmp : stress[jj];
        }
    }
}
// min
} else if(this->stress_read == 2) {
    float _tmp;
    for(int jj=0; jj<6; jj++){
        _tmp = this->buffer->read_float(layerStart+jj);
        if(iLayer == 0){
            stress[jj] = _tmp;
        } else {
            stress[jj] = (_tmp < stress[jj]) ? _tmp : stress[jj];
        }
    }
}
// out
} else if((this->stress_read == 3) & (iLayer == dyna_maxint - 1) ) {
    for(int jj=0; jj<6; jj++)
        stress[jj] = this->buffer->read_float(layerStart+jj);
// mid
} else if((this->stress_read == 4) & (iLayer == dyna_maxint / 2) ) {
    for(int jj=0; jj<6; jj++)
        stress[jj] = this->buffer->read_float(layerStart+jj);
// in
} else if((this->stress_read == 5) & (iLayer == 1) ) {

```

```

    for(int jj=0; jj<6; jj++)
        stress[jj] = this->buffer->read_float(layerStart+jj);
// mean
} else if((this->stress_read == 6)) {

    for(int jj=0; jj<6; jj++)
        stress[jj] += this->buffer->read_float(layerStart+jj);

    if((iLayer == dyna_maxint -1)){
        float _tmp = (float) dyna_maxint;
        for(int jj=0; jj<6; jj++)
            stress[jj] /= _tmp;
    }
}
} // end:stress

// layers: history vars (in/mid/out,mean,min/max)
for(size_t jj=0; jj<this->history_shell_read.size(); ++jj){

    // Skip if over limit
    if(this->history_shell_read[jj] > this->dyna_neips)
        continue;

    // max
    if(this->history_shell_mode[jj] == 1) {
        float _tmp;
        _tmp = this->buffer->read_float(layerStart+6+history_shell_read[jj]);
        if(iLayer == 0){
            history_vars[jj] = _tmp;
        } else {
            history_vars[jj] = (_tmp > history_vars[jj]) ? _tmp :
history_vars[jj];
        }
    }
    // min
    } else if(this->history_shell_mode[jj] == 2) {
        float _tmp;
        _tmp = this->buffer->read_float(layerStart+6+history_shell_read[jj]);
        if(iLayer == 0){
            history_vars[jj] = _tmp;
        } else {
            history_vars[jj] = (_tmp < history_vars[jj]) ? _tmp :
history_vars[jj];
        }
    }
}

```

```

        // out
        } else if((this->history_shell_mode[jj] == 3) & (iLayer ==
dyna_maxint - 1) ) {
            history_vars[jj] = this->buffer-
>read_float(layerStart+6+history_shell_read[jj]);
            // mid
        } else if((this->history_shell_mode[jj] == 4) & (iLayer ==
dyna_maxint / 2) ) {
            history_vars[jj] = this->buffer-
>read_float(layerStart+6+history_shell_read[jj]);
            // in
        } else if((this->history_shell_mode[jj] == 5) & (iLayer == 1) ) {
            history_vars[jj] = this->buffer-
>read_float(layerStart+6+history_shell_read[jj]);
            // mean
        } else if((this->history_shell_mode[jj] == 6)) {

            history_vars[jj] += this->buffer-
>read_float(layerStart+6+history_shell_read[jj]);
            if(iLayer == dyna_maxint-1){
                float _tmp = (float) dyna_maxint;
                history_vars[jj] /= _tmp;
            }

        }
    } // loop:history

} // loop:layers

Element* element = this->db_elements-
>getElementByIndex(SHELL,iElement);

if(dyna_istrn & this->plastic_strain_read)
    element->add_plastic_strain(plastic_strain);
if( this->stress_read )
    element->add_stress(stress);
if(this->history_shell_read.size())
    element->add_history_vars(history_vars,iState);

// strain tensor (in/out,mean,min/max)
if((dyna_istrn == 1) & this->strain_read ){

    vector<float> strain(6);
    int strainStart = (dyna_nv2d >= 45) ? ii+dyna_nv2d - 13 :
ii+dyna_nv2d - 12;

```

```

// max
if(this->strain_read == 1) {
    float _tmp;
    float _tmp2;
    for(int jj=0; jj<6; jj++){
        _tmp = this->buffer->read_float(strainStart+jj);
        _tmp2 = this->buffer->read_float(strainStart+jj+6);
        strain[jj] = (_tmp > _tmp2) ? _tmp : _tmp2;
    }
} else if(this->strain_read == 2) {
    float _tmp;
    float _tmp2;
    for(int jj=0; jj<6; jj++){
        _tmp = this->buffer->read_float(strainStart+jj);
        _tmp2 = this->buffer->read_float(strainStart+jj+6);
        strain[jj] = (_tmp < _tmp2) ? _tmp : _tmp2;
    }
} else if(this->strain_read == 3) {
    for(int jj=0; jj<6; jj++){
        strain[jj] = this->buffer->read_float(strainStart+6+jj);
    }
} else if(this->strain_read == 5) {
    for(int jj=0; jj<6; jj++){
        strain[jj] = this->buffer->read_float(strainStart+jj);
    }
}
// mean/mid
} else if( (this->strain_read == 6) | (this->strain_read == 4) ) {
    for(int jj=0; jj<6; jj++){
        strain[jj] = ( this->buffer->read_float(strainStart+jj)
            +this->buffer->read_float(strainStart+jj+6))/2.f;
    }
}
}
element->add_strain(strain);
}
if( this->energy_read & (dyna_ioshl4)) {
    if((dyna_istrn == 1)){
        if(dyna_nv2d >=45)
            element->add_energy(this->buffer->read_float(ii+dyna_nv2d-1));
    } else {
        element->add_energy(this->buffer->read_float(ii+dyna_nv2d-1));
    }
}
}

iElement++;

```

```

}
string Dplot::get_filepath () {
    return this->filename;
}

vector<float> Dplot::get_timesteps() {
    return this->timesteps;
}
DB_Nodes* Dplot::get_db_nodes(){
    return this->db_nodes;
}
DB_Parts* Dplot::get_db_parts(){
    return this->db_parts;
}
DB_Elements* Dplot::get_db_elements(){
    return this->db_elements;
}
bool Dplot::displacement_is_read(){
    return this->disp_is_read;
}

```

ДОДАТОК Б
Слайди презентації

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА
ПОЛІТЕХНІКА»

Дослідження та розробка засобів
оброблення даних для аналізу
результатів випробувань технічних
об'єктів

Виконав: студент 5 курсу, групи КНТ-219м
Керівник: доцент. Кафедри ІПЗ

Кугай А.Е.
Шипікова О.В..

Рисунок Б.1 – Слайд № 1

Об'єкт дослідження – програмні засоби для обробки даних аналізу результатів випробувань технічних об'єктів

Предмет дослідження – комп'ютерні програми для обробки даних аналізу результатів випробувань технічних об'єктів

Мета роботи – підвищення швидкості обчислень за рахунок створення засобу для оброблення даних та аналізу результатів випробувань технічних об'єктів

Відповідно до мети були поставлені наступні завдання:

- ▶ провести аналіз засобів для оброблення даних та аналізу результатів випробувань технічних об'єктів;
- ▶ провести аналіз існуючих програмних рішень;
- ▶ спроектувати систему вилучення файлів.
- ▶ провести моделювання та розрахунок випробування.

Рисунок Б.2 – Слайд № 2

ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

В автомобілебудуванні з кожним роком все більше уваги приділяється безпеці автомобілів. На сьогоднішній день практично всі моделі автомобілів, особливо популярних виробників, перш ніж надійти в продаж, проходять краш-тести. Краш-тест це випробування автомобілів в спеціально відтворених умовах аварій. Подібні дослідження потребують великих обчислювальних потужностей. Для виконання високопродуктивних розрахунків, а також обробки великих масивів даних, як правило, використовують суперкомп'ютери або обчислювальні кластери. Тому завдання розроблення засобів для обробки та обчислення даних дослідження технічних об'єктів є актуальною.



Рисунок Б.3 – Слайд № 3

Випробування технічних об'єктів

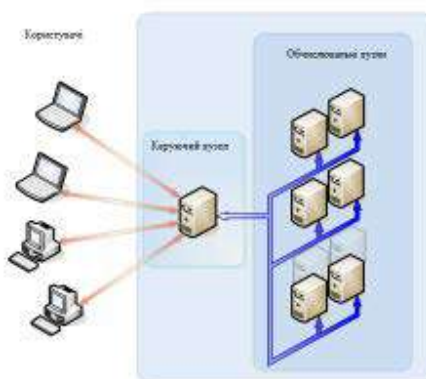


Рисунок 1 – Схема кластера

Рисунок Б.4 – Слайд № 4

Існуючі аналоги системи

СКІТ-1,

СКІТ-2,

СКІТ-3,

СКІТ-4

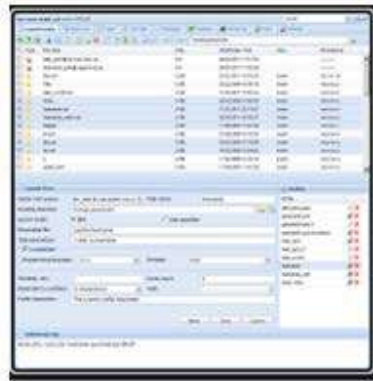


Рисунок Б.5 – Слайд № 5

Програмний засіб для моделювання випробувань

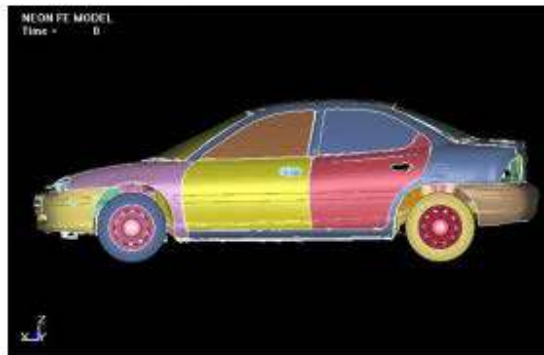


Рисунок 2 – Структурна модель



Рисунок 3 - Модель після удару

Рисунок Б.6 – Слайд № 6

Вибір відкритої інфраструктури для розподілених обчислень

У даних тестах проводився розрахунок з обчислювальною сіткою 1,480,422 на 1,450,225 значень, наступні обчислення таких моделей як Toyota Camry будуть використовувати для обчислень сітку приблизно 1,900,000 на 1,900,000 значень.

Тому було обрано інфраструктуру для розподілених обчислень – BOINC.

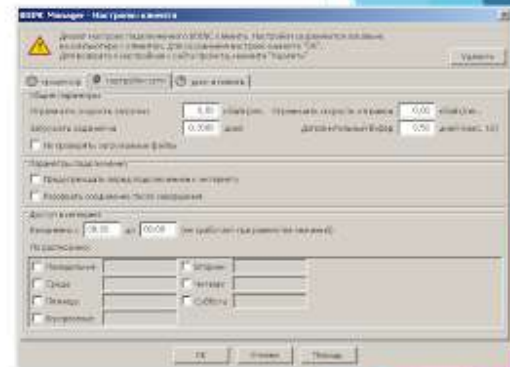


Рисунок 4 - Інтерфейс програми

Рисунок Б.7 – Слайд № 7

Налаштування та тестування кластера



Рисунок 5 - Запуск uname через cluster-fork



Рисунок 6 - Другий запуск uname через cluster-fork

Рисунок Б.8 – Слайд № 8

Проектування в програмному середовищі LS DYN4

Основні дані:

- Кількість деталей - 917;
- Вузлів - 1,480,422;
- Елементів - 1450,225;
- Швидкість - 56, 3 км / год;
- 4324 променевих з'єднання;
- 423 вузлів твердого з'єднання;
- 16 вузлів рухомого з'єднання;
- 2862 точкових зварювання.



Рисунок 7 - Точковий вид

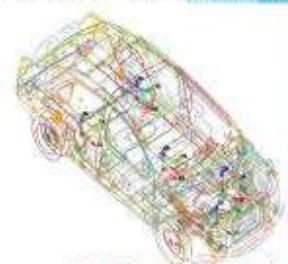


Рисунок 8 - Зображення вузлів, з'єднань

Рисунок Б.9 – Слайд № 9

Зміна прискорення при зіткненні в залежності від часу

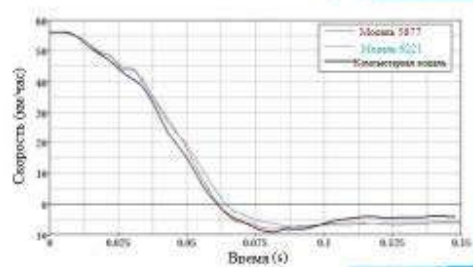
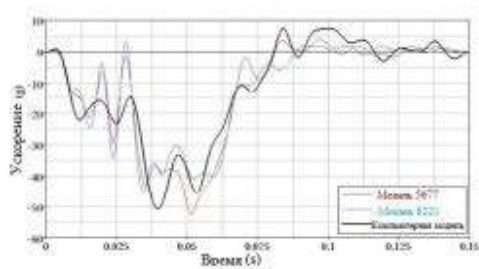


Рисунок 9 - Ліве заднє сидіння

Рисунок Б.10 – Слайд № 10

Зчитувач файлів

Результати повертаються до внутрішніх класів, таких як Node або Element, які можна отримати як за допомогою ідентифікатора, так і з частин. Для елементів оболонки, які мають кілька результатів (напруження та пластична деформація на шар інтеграції та 2х тензор деформації). Для імпорту даних використовуємо наступні бібліотеки:

```
#include "../dyna/dplot.h"
#include "../db/DB_Nodes.h"
#include "../db/DB_Elements.h"
#include "../db/Node.h"
#include "../db/Element.h"
#include <iostream>
#include <vector>
#include <set>
```

```
bool use_femzip=false;
vector<string> variables;
variables.push_back("disp");

Dplot* dplot = new D3plot("filepath/to/dplot", use_femzip, variables);
vector<float> timesteps = dplot->get_timesteps();

variables.push_back("plastic_strain_max");
dplot->read_states(variables); // double reading of disp is caught

Node* node = dplot->get_db_nodes()->get_nodeByID(1);
vector<vector<float>> node_displacement = node->get_disp();

Element* element = dplot->get_elementByID(SHELL,11);
vector<float> elem_plastic_strain = element->get_plastic_strain();
for (auto node : element->get_nodes() )
    cout << "Node:" << node->get_nodeID() << endl;

Part* part = dplot->get_db_parts()->get_part_fromID(13);
set<Element*> part_elems = part->get_elements();
```

Рисунок Б.11 – Слайд № 11

Налаштування розрахунків

Після вибору проекту необхідно налаштувати які ресурси користувач готовий надати. В даному меню користувач вказує коли проекту можна споживати обчислювальну потужність, скільки процесорів можна використовувати, і на скільки відсотків можна їх використовувати.

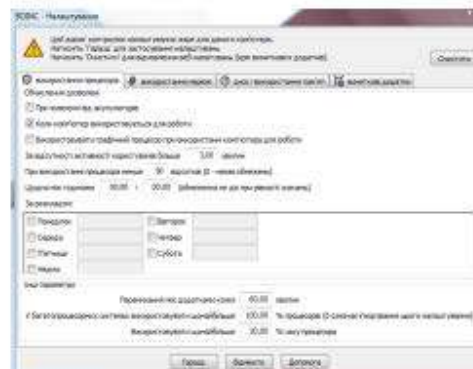


Рисунок 10 – Налаштування

Рисунок Б.12 – Слайд № 12

Результати

Ідеальна обчислювальна потужність зібраного кластера становить 485 Гфлоп / сек, але в цьому кластері можливі втрати рівні 12% від ідеальної потужності. Таким чином, розрахунок на кластері завдання з кількістю вузлів 1,480,422 на 14 50,225 тривав 5 годині 54 хвилини, а на суперкомп'ютері 2 години 40 хвилин.

Зважаючи на те, що запропонований засіб дозволяє економити час на зчитуванні файлів та проводити розрахунки незалежно від сторони ПК, за умови використання 25 робочих станцій, дозволяє проводити розрахунки на 5% швидше ніж на 1 ПК.

Рисунок Б.13 – Слайд № 13

Висновки

У дипломному проєкті був розроблений засіб оброблення даних для аналізу результатів випробувань технічних об'єктів на базі дослідження моделювання безпеки автомобіля.

Мережа включає 25 робочих станцій. Розглянуто перспектива збільшення кількості вузлів.

В ході дипломного проєкту були вирішені наступні завдання:

- проведено порівняльний аналіз процесорів, технологій, інтерфейсів;
- виконано аналіз комп'ютерних архітектур, на підставі якого було розроблено технічне обґрунтування апаратної частини, була обрана архітектура;
- виконано обґрунтування програмних компонентів необхідних для розрахунків і виконана установка;
- виконано проєктування програми для зчитування даних з LS-Dyna;
- Виконано створення кластера з використанням мови програмування Python; та C++
- виконано перевірку з використанням платформи Boinc.

Розроблений засіб дозволяє окремим користувачам легко і швидко взаємодіяти один з одним, а так само виконувати розрахунки для проєкту. Таке рішення дозволяє проводити всі необхідні розрахунки на базі робочих станцій на виробництві. І в цьому випадку відпадає необхідність купувати обчислювальні потужності, що дозволяє істотно заощадити бюджет.

Рисунок Б.14 – Слайд № 14