Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи та засоби моделювання розподілених систем обробки та зберігання даних
(тема)

Виконав:

студент II курсу, групи СПм-22-6

Борисов С.О.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доц. Федорченко В.М.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Борисову Станіславу Олеговичу
(прізвище, ім’я, по батькові)

1. Тема роботи Методи та засоби моделювання розподілених систем обробки та зберігання даних

затверджена наказом по університету від “ 01 ” квітня 2024 р. № 257 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 червня 2024 р.

3. Вхідні дані до роботи
розподілена обробка даних
модель
зберігання даних

4. Перелік питань, що потрібно опрацювати у роботі
Аналіз програмних інструментів імітаційного моделювання грід і хумачних систем
Розробка моделів, методів, алгоритмів і програмних інструментів для синтезу

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 15 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання та аналіз літератури	01.04.2024 – 06.04.2024	
2	Огляд існуючих рішень та методів	07.04.2024 – 12.04.2024	
3	Розробка моделі	13.04.2024 – 18.04.2024	
4	Вибір програмних засобів	19.04.2024 – 25.04.2024	
5	Програмна реалізація	26.04.2024 – 02.05.2024	
6	Аналіз отриманих результатів	03.05.2024 – 16.05.2024	
7	Оформлення записки	17.05.2024 – 14.06.2024	
8	Представлення роботи в ЕК	15.06.2024	

Дата видачі завдання 01 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Федорченко В.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 61 с., 10 рис., 1 дод., 6 джерел.

РОЗПОДІЛЕНА СИСТЕМА, ЗБЕРІГАННЯ ДАНИХ, ОБРОБКА ДАНИХ, МОДЕЛЮВАННЯ.

Метою кваліфікаційної роботи є є аналіз методів та засобів моделювання розподілених систем обробки та зберігання даних.

У ході виконання кваліфікаційної роботи проведено аналіз методів та засобів моделювання розподілених систем обробки та зберігання даних. Досліджено підходи до моделювання розподілених систем зберігання та обробки даних великих обсягів. Розроблено підхід для моделювання систем зберігання та обробки даних з використанням результатів моніторингу. Розроблено програмні засобі для моделювання систем зберігання та обробки даних. Результатом роботи моделі є знайдена величина часу обробки потоку завдань для різних варіантів структури обчислювальної установки та продуктивності окремих її частин, що дозволяє оцінити, як ці фактори впливають на час обробки. Також користувач отримує дані про навантаження на ресурси системи (CPU, RAM, дисковий буфер), час очікування завдань у черзі, пропускну спроможність мережі, дані щодо навантаження на робота стрічкової бібліотеки. Крім того в результаті моделювання можна уточнити які резерви має обчислювальна установка.

ABSTRACT

Master’s thesis: 61 pages, 10 figures, 1appendices, 6 sources.

DISTRIBUTED SYSTEM, DATA STORAGE, DATA PROCESSING, SIMULATION.

The major goal of this thesis is to analyze the methods and means of modeling distributed data processing and storage systems.

In order to the qualification work, an analysis of methods and tools for modeling distributed data processing and storage systems was carried out. Approaches to the modeling of distributed systems of storage and processing of large volumes of data have been studied. An approach has been developed for modeling data storage and processing systems using monitoring results. Software tools for modeling storage and data processing systems have been developed. The result of the model is the found value of the processing time of the flow of tasks for various variants of the structure of the computer installation and the performance of its individual parts, which allows us to assess how these factors affect the processing time. The user also receives data on the load on system resources (CPU, RAM, disk buffer), the waiting time of tasks in the queue, network bandwidth, data on the load on the tape library robot. In addition, as a result of the simulation, it is possible to clarify what reserves the computing facility has.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ОС – операційна система

ПЗ – програмні засоби

ВСТУП

Сучасні системи зберігання та обробки даних – це складні розподілені програмно-апаратні комплекси, побудовані із застосуванням грід та хмарних технологій, що вимагають певного режиму роботи, що змінюється як при збільшенні обсягів даних, що надходять, так і при зміні якості та складу обладнання. Перш, ніж розпочати створення розподіленої IT-інфраструктури, необхідно вирішити, якою буде її архітектура залежно від вартісних факторів та інтенсивності потоків даних. З цією метою необхідно виконати моделювання аналізованої обчислювальної структури, щоб на отриманій динамічній моделі, що враховує реальну специфіку системи і потоків даних, що надходять, вибрати оптимальну архітектуру. Як показав проведений аналіз доступних аналітичних методів моделювання, через обмежені теоретичні передумови вони не можуть бути застосовані для моделювання складних комп'ютерних комплексів багаторівневої архітектури з реальними розподілами вхідних потоків завдань, складною багатопріоритетною дисципліною їх обслуговування та динамічним розподілом. Системи зберігання та обробки даних є складними та багатокомпонентними установками, що включають кластери, а також вузли, реалізовані в хмарній архітектурі, при їх створенні та зміні необхідно використовувати імітаційне моделювання. Під імітаційною моделлю розуміється універсальний засіб дослідження складних систем, що представляє собою логіко-алгоритмічний опис поведінки окремих елементів системи та правил їх взаємодії, що відображають послідовність подій, що виникають у системі, що моделюється.

Імітаційне моделювання грід та хмарних систем дозволяє виявити вузькі місця в архітектурі центрів обробки даних, проводити експерименти зі зміною топології та заміною ресурсів для перевірки запропонованих рішень без безпосереднього втручання у функціонування обчислювального центру,

тестувати алгоритми управління завданнями та розподіл ресурсів за групами користувачів. Найчастіше моделювання застосовують лише на етапі проектування грід та хмарних систем, проте експерименти тривають роками та десятиліттями, при цьому обсяги оброблюваної інформації мають тенденції зростання, тому одночасно з експлуатацією системи відбувається її розвиток, не тільки якісний, а й кількісний. Очевидно, що для досягнення оптимальних результатів моделювання має мати постійний характер протягом усього життєвого циклу експериментів. В даний час процеси моделювання та моніторингу розглядаються як незалежні завдання, не пов'язані між собою. Щоб підвищити точність результатів, необхідно в якості вхідних даних для моделювання використовувати статистику, накопичену під час роботи подібних обчислювальних інфраструктур. Для цього потрібна розробка програмних засобів, що поєднують процеси моделювання та моніторингу систем зберігання та обробки даних великих наукових експериментів.

Метою кваліфікаційної роботи є аналіз методів та засобів моделювання розподілених систем обробки та зберігання даних.

Об’єкт дослідження:системи зберігання та обробки даних.

Завдання:

- дослідження різних підходів до моделювання розподілених систем зберігання та обробки даних великих обсягів;
- розробка підходу для моделювання систем зберігання та обробки даних з використанням результатів моніторингу ;
- розробка програмних засобів для моделювання систем зберігання та обробки даних.

1 АНАЛІЗ ПРОГРАМНИХ ІНСТРУМЕНТІВ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ ГРІД І ХУМАЧНИХ СИСТЕМ

1.1 Проблеми зберігання та обробки даних наукових експериментів

Стрімкий прогрес комп'ютерних технологій, програмних засобів та вибуховий розвиток глобального інформаційного простору, що виник з появою Інтернету, що об'єднав між собою комп'ютерні мережі у всесвітню систему передачі інформації за допомогою інформаційно-обчислювальних ресурсів, – все це ознаменувало вступ людства до нової ери Великих Даних (Big Data).

Майєр-Шенбергер та Кук'єр, автори книги «Великі дані. Революція, яка змінить те, як ми живемо, працюємо і мислимо», характеризують Великі Дані як «масу нових завдань, що стосуються суспільної безпеки, глобальних економічних моделей, недоторканності приватного життя, моральних правил, правових відносин людини, бізнесу та держави» [1]. Раніше збирання та обробка великих обсягів даних були недоступні людині, а статистика, теорія ймовірності, дозволяли уточнювати результати обчислень за рахунок отримання великої кількості точок даних. Big Data виникає тоді, коли для обробки доступні величезні кількості вимірювань. Автори книги приділяють значну увагу розгляду прикладів використання великих обсягів даних, переконуючи, що ми вже живемо у світі Big Data.

У технічній, науковій та, особливо, у соціальних сферах нас оточує безперервний потік великої кількості інформації, що йде з комп'ютерів, мобільних телефонів, передач різних мас медіа та багатьох інших джерел. Тому часто Big Data визначають, як дані, які занадто великі і складні, щоб їх можна було ефективно запам'ятати, передати та проаналізувати стандартними засобами доступних БД та інших систем зберігання, передачі

та обробки. Але, говорячи про Великі Дані, треба розуміти, що це не просто «дуже багато даних».

У 2001 р. компанія META Group, великий аналітик ринку інформаційних технологій, ввела як визначальні характеристики для Великих Даних так звані «три V» [2]:

- обсяг (Volume, у сенсі величини фізичного обсягу);
- швидкість (Velocity, у сенсах як швидкості приросту, і необхідності високошвидкісної обробки та отримання результатів);
- різноманіття (Variety, у сенсі можливості одночасної обробки різних типів структурованих та неструктурованих даних).

Однак, коли загальний потік даних зростає експоненційно, подвоюючися щороку за рахунок революційних технологічних змін, до 2014 р. навіть цю «три V» модель пропонують розширити, додаючи нові та нові «V», включаючи Validity (обґрунтованість, застосовність), Veracity(достовірність), Value (цінність, корисність), і Visibility (оглядність, здатність до візуалізації) і т.д. [3] . Відповідно до 4-ї парадигми наукових досліджень [4] , проведення досліджень, рухомих даними, стає невід'ємною частиною різних галузей науки, економіки, бізнесу. Без забезпечення все новими даними, що є результатом спостережень, вимірювань у природі та суспільстві, розвиток досліджень у різних галузях з інтенсивним використанням даних стає немислимим [5] .

Найяскравішим прикладом Великих Даних є потоки експериментальних даних фізики високих енергій, що надходять з ВАК (LHC, Large Hadron Collider) в ЦЕРН [6] . За час першого запуску ВАК у 2012 році чотири експериментальні установки на ньому ALICE, ATLAS, CMS та LHCb видавали кожну секунду 1 петабайт (10 15 байт) даних. Запам'ятати таку кількість даних було неможливо на жодній з обчислювальних систем. Тому після надшвидкої складної електронної передобробки в ЦЕРН виконувалася їхня первинна реконструкція в комп'ютерному центрі обробки з багатьох тисяч процесорів. Але навіть після скорочення обсягу

експериментальних даних у мільйон разів, для цих чотирьох великих експериментів потрібно зберігати, що надходять на рік 25 петабайт даних, у спеціальних роботизованих стрічкових сховищах. Копії цих даних підлягають передачі до сотень фізичних центрів у 36 країнах світу, об'єднаних у Всесвітню мережу розподілених обчислень - Worldwide LHC Computing Grid (WLCG). Порівняння ВАК за загальним обсягом даних, що обробляються, з тим, що виконується в соціальних мережах, пошукових системах, різних галузях бізнесу, медицини, кліматичних прогнозів, наочно показує, що дослідження в ЦЕРНі йдуть в умовах Великих Даних.

Таким чином, розвиток наукових досліджень у фізиці високих енергій та інших напрямах людської діяльності вимагають спільної роботи багатьох організацій з обробки великого обсягу даних у відносно короткий термін. Для цього створюються системи розподіленого зберігання та обробки даних, які здатні передавати, обробляти та зберігати величезні масиви даних із застосуванням грід та хмарних технологій.

1.2 Розподілені системи зберігання та обробки даних

1.2.1 Грід системи

Термін "грид-обчислення" з'явився на початку 1990 років в "The Grid: Blueprint for New Computing Infrastructure" під редакцією Яна Фостера і Карла Кессельмана, як метафора, що демонструє можливість простого доступу до обчислювальних ресурсів, як і до електричної мережі (power grid). «Грид – географічно розподілена інфраструктура, що поєднує безліч ресурсів різних типів (процесори, довгострокова та оперативна пам'ять, сховища та БД, мережі), доступ до яких користувач може отримати з будь-якої точки, незалежно від місця їх розташування. Грід передбачає колективний режим доступу до ресурсів і пов'язаних з ними послуг в рамках глобально розподілених віртуальних організацій, що складаються з

підприємств і окремих фахівців, що спільно використовують спільні ресурси. У кожній віртуальній організації є своя власна політика поведінки її учасників, які повинні дотримуватись встановлених правил. Віртуальна організація може утворюватися динамічно та мати обмежений час існування» [7].

В.В. Коріньков у своїй статті «Грид-технології: статус і перспективи» [8] перераховує основні додатки грід, до яких належать:

- складне моделювання на віддалених суперкомп'ютерах;
- спільна візуалізація великих наборів наукових даних;
- розподілена обробка з метою аналізу даних;
- з'єднання наукового інструментарію з віддаленими комп'ютерами та архівами даних.

Також серед основних напрямів використання грід автор статті виділяє:

- організацію ефективного використання ресурсів для невеликих завдань, з утилізацією комп'ютерних ресурсів, що тимчасово простоюють;
- обчислення із залученням великих обсягів географічно
- розподілених даних;
- колективні обчислення, у яких одночасно беруть участь користувачі різних організацій.

Таким чином, грід служить універсальною ефективною інфраструктурою для розподілених обчислень та обробки даних. І якщо спочатку технології грід використовувалися для наукових та інженерних додатків, то тепер вони застосовуються для вирішення різних завдань у державному управлінні, медицині, промисловості, бізнесі.

Наймасштабнішим науковим проектом останніх років є створення ЦЕРН БАК. Дуже великий обсяг даних, що надходять з детектора, вимагає використання грід системи для їх обробки. Ця система має кілька рівнів організації та є розподіленою моделлю зберігання та обробки даних. Суть розподіленої моделі полягає в тому, що весь обсяг інформації з детекторів ВАК після обробки в реальному часі та первинної реконструкції повинен

спрямовуватися для подальшої обробки та аналізу до регіональних центрів.

На урочистості 4 липня 2012 р. з приводу отримання нобелівської премії за відкриття бозона Хіггса директор ЦЕРН Рольф Хойєр прямо назвав грід-технології одним із трьох стовпів успіху (поряд із прискорювачем ВАК та фізичними установками) [10].

1.2.2 Хмарні засоби зберігання даних та обчислень

«Хмарні обчислення – це модель надання зручного мережного доступу в режимі «на вимогу» до колективно використовуваного набору обчислювальних ресурсів, що настроюються (наприклад, мереж, серверів, сховищ даних, додатків і сервісів), які користувач може оперативно задіяти під свої завдання і вивільняти при зведенні до мінімуму кількості взаємодій із постачальником послуги чи власних управлінських зусиль. Ця модель спрямована на підвищення доступності обчислювальних ресурсів» [11] .

Споживач, коли це необхідно, може самостійно використовувати обчислювальні можливості, такі як серверний час або мережеве сховище даних, в автоматичному режимі, без взаємодії з персоналом постачальника послуг. Всі можливості доступні через мережу на основі стандартних механізмів, що забезпечує використання тонких і товстих клієнтських платформ, таких як мобільних телефонів, ноутбуків.

Постачальник об'єднує свої обчислювальні ресурси в пул обслуговування великої кількості споживачів. Різні фізичні та віртуальні ресурси динамічно розподіляються та перерозподіляються відповідно до потреб користувачів. Виникає відчуття незалежності від місця розташування, коли замовник не знає і не контролює, де конкретно знаходяться обчислювальні ресурси, якими він користується.

Обчислювальні можливості можуть швидко та гнучко резервуватися (часто автоматично) для оперативного масштабування під завдання замовника, а також швидко звільнятися. З точки зору споживача, доступні

можливості часто виглядають нічим не обмеженими і можуть бути придбані в будь-якій кількості в будь-який час. Хмарні системи автоматично контролюють і оптимізують використання ресурсів через вимірювання деяких параметрів, таких як розмір сховища даних, обчислювальна потужність, пропускна здатність, кількість активних записів користувача.

Таким чином, хмарні технології забезпечують мережевий доступ до обчислювальних, програмних та інформаційних ресурсів (мереж передачі даних, серверів, пристроїв зберігання, сервісів і додатків), що конфігуруються відповідно до оперативних запитів. Вони дозволяють значно скоротити витрати на IT-інфраструктуру, задовольняти потреби, що динамічно змінюються, в ресурсах і т.д.

1.2.3 Гібридні розподілені системи зберігання та обробки даних

Для широкого спектру завдань у різних галузях науки є актуальним скорочення часу їх виконання, а також підвищення ефективності використання ресурсів. Одним із рішень є синтез хмарних та грід-технологій. Як було показано у дисертації Н.А. Кутовського [12] підвищення ефективності використання комп'ютерних ресурсів досягається при розміщенні грід-сервісів на віртуальних машинах в хмарному середовищі, а зменшення часу виконання – за рахунок вирішення завдань на окремих спеціалізованих комплексах. «Таким чином, є актуальним розробка тметодів створення багатофункціональних гетерогенних комплексів для вирішення широкого класу завдань у галузі фізики високих енергій, що дозволяють скоротити час вирішення цих завдань та підвищити ефективність використання ресурсів».

Прикладом вже наявної технології, що реалізує синтез хмарних та грід-технологій для роботи з Великими Даними, є система PanDA (Production and Distributed Analysis – обробка даних та розподілений аналіз) експерименту ATLAS на LHC.

У 2018 р. під час другого запуску ВАК, навіть після відсіву 99% даних, з детекторів ВАК було отримано 88 ПБ даних [14], що потребувало значного збільшення обчислювальних потужностей та ресурсів зберігання даних [16]. Слід зазначити, що в 2021 р. чекає третій запуск ВАК після його суттєвої модернізації, де очікується дворазове збільшення даних, що збираються, і неминучий перехід до гібридних IT-інфраструктур. Це необхідно для потенційно нової фізики, але стикається з новими серйозними вимогами, а саме:

- значне збільшення обчислювальних потужностей та мережевих ресурсів зберігання даних;
- необхідність доступу до даних з грід та хмар;
- активне використання розподілених паралельних обчислень;
- вдосконалення кодів програм аналізу та моделювання.

На сьогоднішній день грід та хмарні технології активно застосовуються як державними організаціями у сфері управління, оборони, комунальних послуг, так і приватними компаніями, наприклад, фінансовими та енергетичними. Такі системи використовуються, наприклад, для обробки та зберігання даних з експериментів фізики високих енергій, де прискорювачі частинок виробляють обсяг даних до сотень петабайт на рік. Слід підкреслити, що розробка таких складних систем збору, передачі та розподіленої обробки надвеликих обсягів інформації потребує великих попередніх досліджень щодо вибору їх оптимальної структури з урахуванням вартості передбачуваних ресурсів та їх завантаження.

1.3 Аналіз інструментів моделювання

Якщо говорити про моделювання як про метод наукового пізнання, то моделювання це один із методів пізнання, що полягає в описі реальних об'єктів і явищ за допомогою інших об'єктів і явищ або за допомогою абстрактного опису у вигляді зображення, рівняння, формули, програми для

комп'ютера. Таким чином, модель – це уявлення реального об'єкта у формі, відмінній від його реального втілення.

За способами моделювання розрізняють:

- фізичне, натурне моделювання;
- структурно-функціональна (за допомогою мови, блок-схеми, графіка, карти);
- математичне (за допомогою аналітичного опису: рівнянь та формул);
- імітаційне моделювання для тих випадків, коли аналітичний опис неможливий через складність описуваного явища. Оскільки імітаційне моделювання виникло і може бути виконане лише на комп'ютері, його часто називають комп'ютерним моделюванням, що, зрештою, звужує опис цього способу моделювання, т.к. В даний час на комп'ютері реалізуються і аналітичні та різні числові моделі явищ.

Як цілі моделювання, можна назвати: осмислення, вивчення дійсності; спілкування; навчання, тренаж.

Функції моделювання:

- конструювання та перевірка моделі;
- експериментування з моделлю вивчення поставленої завдання, вибору оптимальної стратегії її вирішення чи передбачення поведінки досліджуваного процесу у майбутньому.

Моделі класифікують:

- за вищезазначеними способами моделювання;
- по відношенню до часу: статичні чи динамічні моделі;
- до випадковості: детерміністські або стохастичні моделі [15].

Якщо тепер вибирати спосіб моделювання, найбільш адекватний задачі, що вирішується, то очевидно, що фізичне моделювання нам не підходить. При фізичному моделюванні досліджувана система замінюється відповідної їй інший матеріальної системою, яка відтворює властивості системи, що вивчається, зі збереженням їх фізичної природи. Можливості фізичного моделювання обмежені. Воно дозволяє вирішувати окремі

завдання при заданні невеликої кількості поєднань досліджуваних параметрів системи. При фізичному моделюванні практично неможливо перевірити роботу системи різних варіантів. Перевірка практично близько десятка різних типів умов пов'язані з великими зусиллями, тимчасовими і чималими матеріальними затратами. У багатьох областях досліджень фізичний експеримент неможливий, тому що він або заборонений (вивчення здоров'я людини), або занадто небезпечний (вивчення екологічних явищ), або просто неможливий (вивчення астрофізичних явищ).

Тому в багатьох випадках кращим є використання математичного моделювання. Математична модель являє собою сукупність співвідношень (формул, рівнянь, нерівностей, логічних умов), що визначають процес зміни стану системи в залежності від її параметрів, вхідних сигналів, початкових умов та часу. Під математичними моделями розуміють основні закономірності та зв'язки, властиві досліджуваному явищу. Це може бути формули чи рівняння, набори правил чи угод, виражені у математичної формі [17].

В даний час широко застосовується два види математичного моделювання: аналітичне та імітаційне. При аналітичному моделюванні вивчаються математичні (абстрактні) моделі реального об'єкта у вигляді алгебраїчних, диференціальних та інших рівнянь, а також що передбачають здійснення однозначної обчислювальної процедури, що призводить до їх точного вирішення. При імітаційному моделюванні досліджуються математичні моделі у вигляді алгоритму(ів), що відтворює функціонування системи, що досліджується, шляхом послідовного виконання великої кількості елементарних операцій [18].

Аналітичне моделювання дозволяє отримувати більш точне рішення, оскільки головним завданням є вирішення рівнянь для отримання теоретичних результатів та їх зіставлення з практикою. Але з допомогою цього виду моделювання дуже складно провести повне дослідження процесу функціонування складної системи, вивчити її властивості.

Вивчати складні системи стало можливим у розвитку інформаційних технологій, коли комп'ютери стали використовуватиме моделювання процесів функціонування системи. У цьому випадку були алгоритм та програма, а математична модель у її класичному вигляді практично була відсутня або передбачалося, що математичною моделлю є одне з аналітичних уявлень. Цей напрямок отримав назву імітаційного моделювання. Такі моделі є комп'ютерною програмою, яка крок за кроком відтворює події, що відбуваються в реальній системі. Перевагою імітаційних моделей є можливість заміни процесу зміни подій у досліджуваній системі у реальному масштабі часу на прискорений процес зміни подій у темпі роботи програми.

Говорячи про те, яку технологію застосувати для моделювання грід і хмарних систем, необхідно враховувати, що можливість застосування аналітичних моделей для завдань обмежена. Існує кілька підходів при аналітичному моделюванні грід та хмарних систем, які можна згрупувати у два типи:

- система розглядається, як багатоканальна система масового обслуговування, зі станами, керованими Марківським процесом, з обмеженнями на розподіл вхідних потоків та на дисципліни обслуговування, викликаними теоретичними передумовами;

- система розглядається як динамічна стохастична мережа, що описується системами рівнянь, що дозволяють враховувати, як маршрутизацію, так і розподіл ресурсів у мережі, причому вивченню підлягають рівноважні та нерівноважні стани мережі [19] .

Обидва підходи видають результат моделювання, як правило, у вигляді асимптотичних розподілів і через обмежені теоретичні передумови не можуть бути застосовані для моделювання конкретних складних комп'ютерних мереж багаторівневої архітектури з реальними розподілами вхідних потоків завдань, складною багатопріоритетною дисципліною їх обслуговування та динамічним розподілом.

Тому для моделювання процесів керування розподіленими даними доцільніше використовувати імітаційне моделювання.

Імітаційна модель відтворює поведінку складної динамічної системи взаємодіючих компонентів, її логіко-алгоритмічний опис визначається такими ознаками:

- об'єкт моделювання – складна неоднорідна система;
- для моделюється системи характерна наявність випадкових факторів;
- система динамічна та процес її розвитку в часі має бути описаний.

Стан кожного компонента системи, що моделюється, описується набором параметрів. Програма, що реалізує імітаційну модель, відображає зміну стану системи, виконуючи моделювання в покроковому режимі. Значення параметрів системи змінюються за кроками часу або в послідовності подій, що відбуваються в системі. На сьогоднішній день існують різні програмні інструменти імітаційного моделювання грід та хмарних систем, огляд яких представлений нижче в цьому розділі.

1.3.1 Комп'ютерне моделювання

Технології комп'ютерного моделювання широко використовують у час. Доцільність модельного забезпечення складних технічних розробок та наукових досліджень сьогодні не викликає жодних сумнівів. У майбутньому роль та значення комп'ютерного моделювання, безумовно, значно зросте.

Сучасне комп'ютерне моделювання постає як засіб спілкування людей (обмін інформаційними, комп'ютерними моделями та програмами), осмислення та пізнання явищ навколишнього світу (комп'ютерні моделі сонячної системи, атома тощо), навчання та тренування (тренажери), оптимізації (підбір параметрів) [20] .

У звіті про науково-дослідну роботу співробітників ЛІТ ОІЯД проект з гранту Міністерства освіти і науки [21] на тему «Модель розподіленої системи колективного користування для збору, передачі та обробки

надвеликих обсягів інформації на основі технології грід для прискорювального комплексу НІКА» наведено докладний аналітичний огляд інформаційних джерел та аналіз ефективності існуючих рішень комп'ютерних моделей. Зі звіту можна зробити висновок про те, що комп'ютерне моделювання є одним з ефективних методів вивчення складних систем, оскільки комп'ютерні моделі простіше і зручніше досліджувати в силу їх можливості проводити обчислювальні експерименти, які в порівнянні з реальним експериментом утруднені через фінансові та фізичні перешкоди . або можуть дати непередбачуваний результат. Логічність і формалізованість комп'ютерних моделей дозволяє виявити основні фактори, що визначають властивості досліджуваного об'єкта-оригіналу (або цілого класу об'єктів), зокрема, дослідити відгук фізичної системи, що моделюється, на зміни її параметрів і початкових умов.

Розвиток інформаційних технологій призвело до того, що комп'ютери стали використовувати для моделювання процесів функціонування системи, причому в цьому випадку були алгоритм та програма, а математична модель у її класичному вигляді практично була відсутня або передбачалося, що математичною моделлю є одне з аналітичних уявлень. Цей напрямок отримав назву імітаційного моделювання. Такі моделі є комп'ютерною програмою, яка крок за кроком відтворює події, що відбуваються в реальній системі. Перевагою імітаційних моделей є можливість заміни процесу зміни подій у досліджуваній системі у реальному масштабі часу на прискорений процес зміни подій у темпі роботи програми.

При створенні грід та хмарних систем імітаційне моделювання найчастіше застосовують лише на етапі проектування. У роботі [22] описано програму моделювання грід структури. Для запуску програми потрібно задати склад і топологію центрів обробки структури, що моделюється, структуру, а також розподіл ресурсів між завданнями. Після цього програма виконує імітаційне моделювання процесів проходження згенерованого набору завдань через цю структуру грід. Як результати обчислюються

тимчасові оцінки параметрів потоку завдань, що шукаються. Моделювання системи дозволяє відповісти на низку питань щодо підбору найкращих параметрів системи. При створенні розподіленої системи потрібно ухвалити рішення щодо архітектури інфраструктури, кількості ресурсних центрів, обсягу необхідних ресурсів.

Проте експерименти продовжуються роками та десятиліттями. У зв'язку з якісним розвитком системи необхідно забезпечити достатню пропускну здатність, вирішити проблеми збереження даних (стійкість до пошкоджень та видалень) протягом усього життєвого циклу проекту, забезпечити розподіл ресурсів між різними групами користувачів, вибрати алгоритми обробки та запуску завдань та багато іншого.

Але водночас з експлуатацією відбувається як якісний, а й кількісний розвиток системи. Таким чином, навіть при значних зусиллях, вкладених на етапі проектування у розуміння конфігурації систем та їх кількісних характеристик, неможливо розвивати систему без додаткових досліджень. Розробники та експлуатуючі організації стикаються з проблемою прогнозування поведінки системи після проведення запланованих модифікацій [23].

Таким чином, потрібне створення методології та програмного оточення, що дозволяє не тільки моделювати системи на постійній основі, а й прогнозувати поведінку системи за її змін. Як дані для такого прогнозу можна використовувати статистику, накопичену під час роботи програми моніторингу системи, що моделюється. При цьому необхідно врахувати, що модель має розглядатися як невід'ємна частина системи обробки даних, а дані моніторингу як вхідні для моделювання. Це дозволить приймати обґрунтованіші проектні рішення при розвитку системи. Також об'єднавши моделювання та моніторинг у рамках одного програмного пакета, можна досягти суттєвого зниження експлуатаційних витрат та вкладень у збільшення потужності з метою збереження швидкості отримання результату експериментів, при постійному підвищенні потоку даних.

1.3.2 Системи моделювання грід

Однією з актуальних завдань на даний час є ефективне управління ресурсами зберігання та розподілом обчислювальних ресурсів у розподіленому середовищі. В даний час існують різні системи моделювання грід. Здебільшого ці програми моделюють обчислювальні грід системи. Це дозволяє вивчати різні алгоритми запуску завдань, політики резервування ресурсів. Система моделювання може бути використана для оцінки ефективності розподіленого обчислювального середовища в різних ситуаціях, наприклад: при зміні навантаження: кількості завдань, їх розмірності, пріоритету, періоду надходження і т.д.; у разі відключення частини обчислювальних ресурсів або додавання нових ресурсів; зі збільшенням кількості переданих даних; при виході з експлуатації частини комунікаційних каналів.

При цьому оцінка ефективності управління може проводитись за такими найбільш популярними критеріями:

- мінімізація середнього часу очікування завдання у черзі;
- мінімізація максимального часу виконання групи завдань;
- максимізація пропускної спроможності - числа завершених завдань за одиницю часу;
- мінімізація простоїв процесорів тощо.

Витрати на проектування та розвиток грід-систем можуть бути значно зменшені у разі використання ефективних методів моделювання.

Грід-системи, орієнтовані зберігання та обробку великих масивів інформації, називаються DataGrid. DataGrid технологія затребувана в наукових колах: астрономія, моделювання білка, фізика високих енергій. Експерименти в цих областях генерують велику кількість даних, які мають бути загальнодоступними для вчених для обробки та аналізу. У пакетах моделювання Brick [24], OptorSim [25] і GridSim [26] зроблено спробу

моделювання DataGrid систем, що може бути основою подальших розробок у цій галузі.

Система моделювання Bricks.

Система моделювання Bricks призначена для моделювання клієнт-серверної архітектури як глобальної обчислювальної системи. У ній передбачається централізоване глобальне планування.

Bricks – система моделювання, призначена для дослідження алгоритмів планувальників завдань та пропускної спроможності каналів. Творці системи керувалися ідеєю, що грид складається з користувачів, які запускають мережі завдання на загальні ресурси. Програма була реалізована Java.

Архітектура.

Система Bricks реалізована як взаємодія двох головних компонент: модуля моделювання обчислювального грід-середовища та модуля планування.

Передбачається, що обчислювальне середовище – мережа, що складається з однорідних елементів (Host). Host включає: Клієнта (Client), Сервер (Server) та об'єкт типу Диск (Disk). Об'єкт Client представляє машину користувача, яка генерує завдання для виконання на ресурсі. Об'єкт Server характеризується продуктивністю, робочим навантаженням та його розподілом протягом часу. Об'єкт Network об'єднує клієнтів та сервера та характеризується пропусканою здатністю, навантаженням та фоновим трафіком. Різні сценарії можуть бути використані до об'єктів. Завдання характеризуються необхідною кількістю операцій. За замовчуванням виконується на одному ресурсі.

Сервер та мережеві ресурси працюють із чергами завдань. Сервер обробляє заплановані завдання методом FCFS (First-come, First-served). Два способи обробки черг реалізовані в Bricks: QueueFCFS та QueueT imeSharing.

Топологія мережі має бути визначена користувачем конфігураційному файлі.

Bricks включає засоби моніторингу, прогнозування і планування роботи системи. Засобами моніторингу проводиться контроль працездатності ресурсів та збереження цих результатів у базі даних (ResourceDB). Зокрема, NetworkMonitor фіксує пропускну здатність та час очікування, ServerMonitor – продуктивність, завантаженість та працездатність серверів. Модуль прогнозування визначає використання та працездатність мережевих ресурсів та серверів. Планувальник розміщує нове завдання на відповідний сервер, беручи до уваги інформацію про стан ресурсу, отриману з ResourceDB та модуля прогнозування.

Функціонування системи.

Перед запуском моделі користувач повинен визначити частоту надходження завдань від клієнта та обсяги завдань. Робота системи починається з аналізу файлів конфігурації.

Коли нове завдання надходить від Клієнта, воно обробляється Планувальником, відповідальним за розміщення завдання з урахуванням його особливостей. Планувальник запитує інформацію в модулі Прогнозування про можливість використання того чи іншого ресурсу та розміщує завдання на відповідний Сервер. Сервер обробляє завдання по черзі. Коли Сервер завершив обробку завдання, результати надсилаються Клієнту.

Статистика експерименту.

Вся інформація про стан ресурсів та процес моделювання доступна. Вона включає: смугу пропускання мережі та час очікування, пропускну здатність сервера та розміри черг, поточне використання центрального процесора та середнє завантаження, розмір завдань, розмір їх відповіді, час, протягом якого завдання було розпочато та виконано та багато іншого.

Копіювання даних.

Структура Bricks дозволяє моделювати копіювання даних (реплікації). Реплікація даних та автоматичне розподілення завдань за ресурсами гарантує баланс навантаження та полегшує доступ до даних усім користувачам.

Охоплення всіх часових поясів також полегшує цілодобовий моніторинг та підтримку.

Копіювання даних може здійснюватися згідно з централізованою або ієрархічною моделлю. У централізованій моделі всі дані, і всі завдання, що мають справу з цими даними, зберігаються і обробляються на єдиному сайті, у якого, як передбачається, є достатні ресурси для обробки та відповідна ємність пристроїв, що запам'ятовують. Ієрархічна модель передбачає обробку завдань, коли зі збільшенням завантаження сервера, частина даних копіюється ресурси нижнього рівня.

ReplicaManager виконує алгоритм копіювання файлів та збирання інформації про дискові ресурси. Щоб уникнути нестачі дискового простору, Bricks стежить за вільним простором на дискових ресурсах і видаляє дані, коли це необхідно. Критерієм видалення даних є ставлення використаного дискового простору до всього обсягу. Цей поріг визначає користувач конфігураційного файлу. Коли граничне значення перевищено, відбувається видалення файлів.

Система моделювання OptorSim.

OptorSim створювався в рамках європейського проекту European DataGrid (EDG) як інструмент для моделювання DataGrid. У додатках, які працюють із великими обсягами даних, важливо як уникнути втрати даних, а й організувати оптимальний доступом до них. OptorSim - пакет моделювання DataGrid, реалізований мовою Java, дозволяє оцінювати різні алгоритми оптимізації та стратегії копіювання. Вихідний код широко доступний.

Поведінка Користувача (User) визначається конфігураційному файлі і відрізняється розподілом чи частотою, з якою вони запускають завдання на RB. Так, за сценарієм SimpleUsers, користувачі запускають завдання згідно з рівномірним розподілом (час простою між запусками визначається в конфігураційному файлі). RandomWaitUsers моделює ситуацію, коли користувачі відпочивають між запусками випадковий проміжок часу. Клас CMSDC04Users використовує гауссівський розподіл.

Файл даних – об'єкт, що містить інформацію про ім'я файлу, розмір, чи є файл копією або майстер файлом. Майстер файл - оригінальна копія даних, яка не може бути видалена. Набір даних (Dataset) - кілька файлів, які мають спільні властивості. Кожне завдання може містити один або декілька наборів даних. Параметри доступу (access patterns) визначають порядок обробки файлів завдання.

Функціонування системи.

На рисунку 1.1 показано алгоритм роботи ResourceBroker. Користувач запускає нові завдання, які RB обробляє по черзі. Завдання має бути відправлене для виконання на Сайт, який має доступ до всіх файлів у завданні. Коли RB знаходить невиконану роботу, то виконується пошук Сайту з відповідним CE. Якщо знайдений CE є вільним, завдання буде спрямоване на Сайт і додано в чергу на обробку на CE.

Далі CE моделює запуск завдання. Для кожного файлу в задачі визначається "найкраще" розташування файлу для копіювання, згідно з алгоритмом оптимізації. Як тільки цей “кращий” файл було знайдено, FileTransfer моделює передачу даних через мережу.

Коли всі файли отримані, завдання вважається виконаним. На рисунку 1.1 представлена діаграма послідовності дій під час виконання завдання. Файли копіюються один за одним. Якщо кілька обчислювальних елементів копіюють файли по одній мережі, пропускна здатність ділитися між РЄ. Таким чином, кожен наступний файл повинен чекати, доки попередній файл буде переданий.

Статистика експерименту.

Статистика в OptorSim представлена як у кожному елементу окремо, і по роботі моделі. Звітна інформація доступна у вигляді таблиць, графіків та діаграм. Для SE можна отримати інформацію про обсяг дисків та їх завантаження. Є можливість отримати статистику за часом виконання завдань, а також відсоткове співвідношення часу використання CE на час простою, кількість віддалених та локальних запитів до файлу, а також

кількість переданих файлів. Цікавим є показник ефективності використання мережі. Ефективність використання мережі розраховується як відношення кількості переданих файлів (віддалений доступ або реплікація файлів) до запитуваних файлів. Цей показник допомагає вирішити питання – чи слід змінювати стратегію оптимізації та застосувати алгоритм, який поміщає файли на “правильні” Сайти у разі низької ефективності.

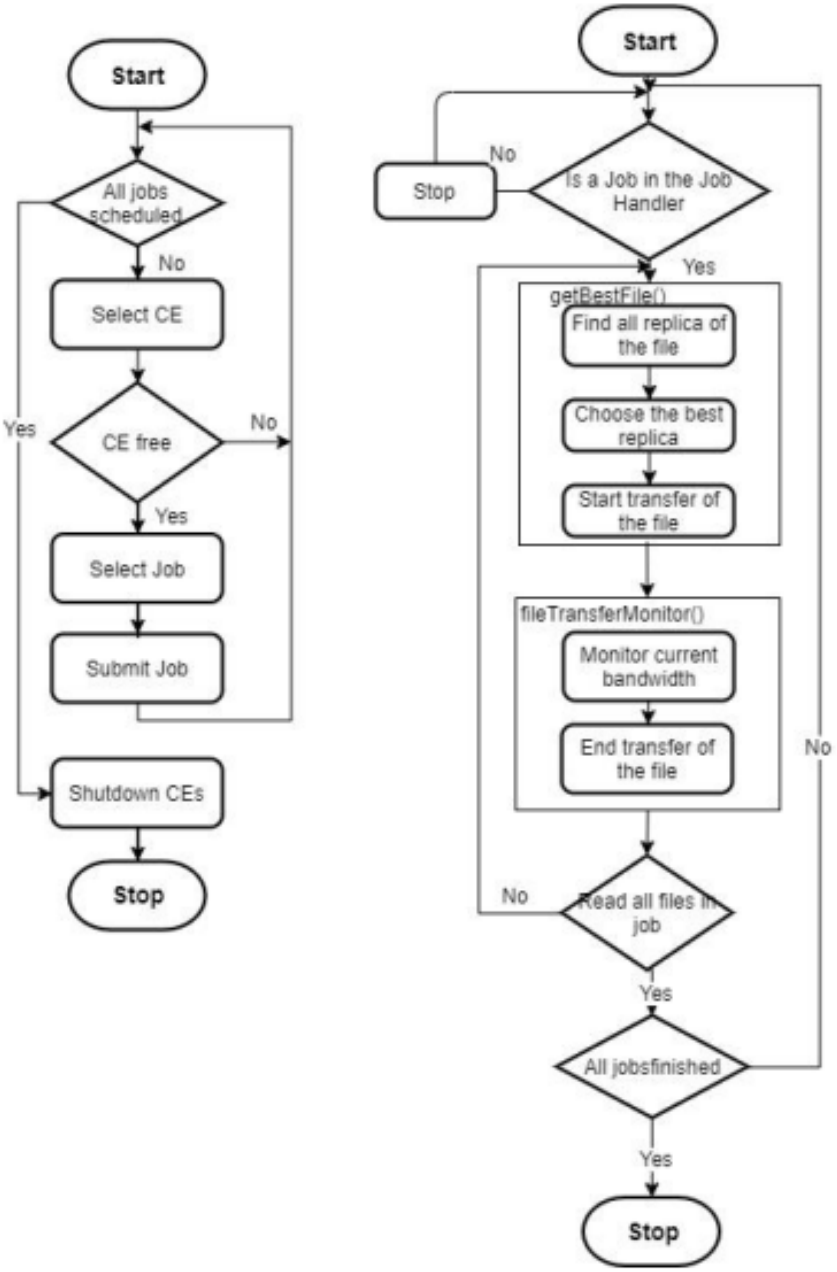

Рисунок 1.1 – Алгоритм роботи RB та CE

Статистика експерименту. Статистика в OptorSim представлена як у кожному елементу окремо, і по Інтерфейсу.

Однією з переваг пакету є наявність графічного інтерфейсу для візуального відображення моделі. Відповідно до рисунка 1.2, графічний інтерфейс дозволяє користувачеві відстежувати стан моделі, навіть під час виконання моделювання. Практичний інтерес представляє закладка Statistics, де поточна інформація представлена у табличній формі. Закладка Logical View показує граф, де відображаються поточні передачі файлів між Сайтами. Детальну інформацію наведено в посібнику користувача.

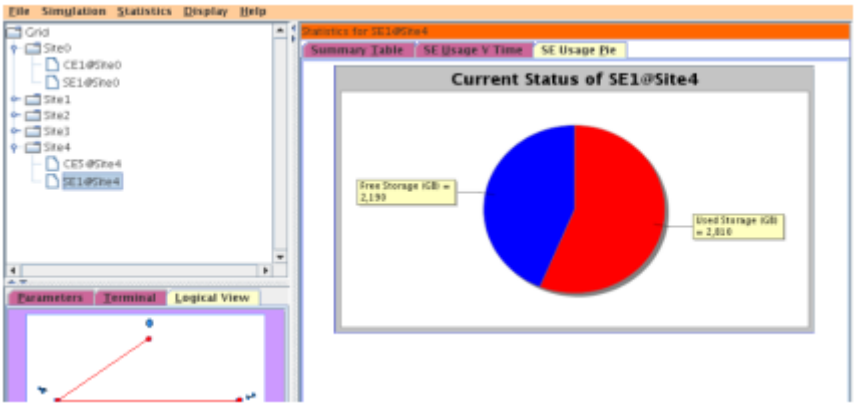

Рисунок 1.2 – Графічний інтерфейс OptorSim [28]

Система моделювання GridSim.

Проект GridSim розробляється групою дослідників у лабораторії з вивчення хмарних та розподілених обчислень відділу інформатики та комп'ютерних обчислень в університеті Мельбурна, Австралія.

Ця програма дозволяє користувачам моделювати роботу грід-системи за різних конфігурацій. Вона надає можливість моделювати поведінку користувачів грід-системи, обчислювальних ресурсів та брокерів ресурсів (планувальників). GridSim може бути використаний дослідниками, які розробляють та підвищують ефективність існуючих алгоритмів планування завдань на обчислювальних кластерах. За допомогою GridSim можна

проводити експерименти, що відтворюються, які складно реалізувати в цьому оточенні динамічних грід-систем.

Основні можливості GridSim полягають у наступному:

- моделювання різних характеристик ресурсів грід-середовища;
- моделювання різних політик планування завдань на вузлах обчислювальних кластерів – як реалізованих (FCFS, Easy Backfill, Conservative Backfill), і розроблених користувачами алгоритмів;
- використання даних про завантаження реальних кластерів щодо експериментів;
- підтримці механізму аукціону для планування завдань;
- моделювання різних конфігурацій обчислювальної мережі грід-системи для різних топологій;
- моделювання регіональних компонентів грід, інформаційних сервісів.

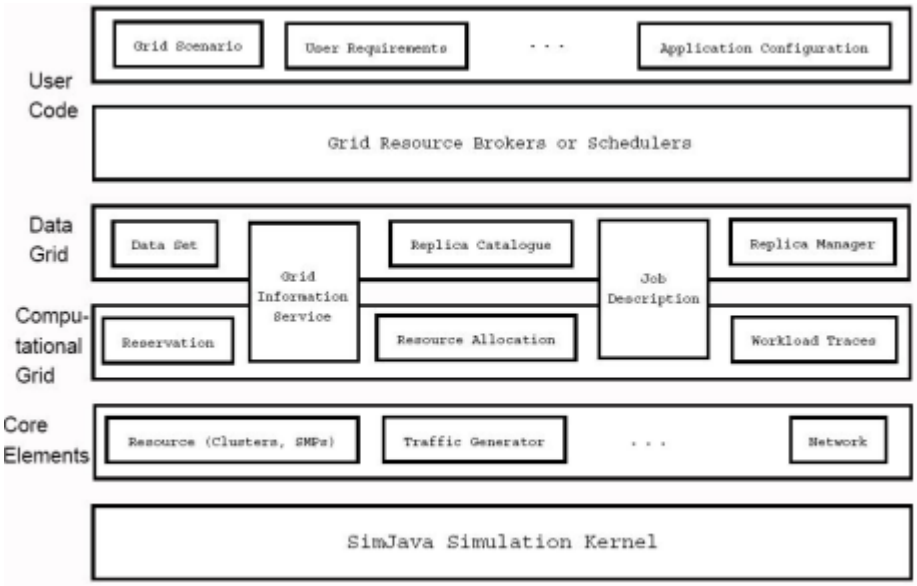

Рисунок 1.3 – Архітектура GridSim

GridSim проектувався як багаторівнева система моделювання, можливості якої можуть бути легко та значно розширені. Архітектура

GridSim представлена на рисунку 1.3 [29]. GridSim заснований на SimJava – бібліотеці для створення дискретно-подійних процесів, реалізованому мовою Java. Тому SimJava займається обробкою внутрішніх подій та взаємодії компонентів GridSim. Всі компоненти GridSim взаємодіють між собою через надсилання повідомлень, визначених у SimJava.

Другий шар моделює основні компоненти розподіленої інфраструктури, а саме, грид-ресурси, наприклад, кластери, сховища даних та мережеві з'єднання. Ці компоненти є істотними при створенні моделей за допомогою GridSim.

Третій та четвертий шар концентруються на моделюванні служб, специфічних для обчислень та технології DataGrid. Деякі служби надають функціональність, загальну для всіх видів грід-систем, наприклад, інформацію про доступність ресурсів системи та управління призначенням (плануванням) завдань. Для випадку DataGrid управління завданнями також включає управління передачею інформації між вузлами зберігання даних і обчислювальними вузлами. Служби керування файлами та даними також реалізовані особливим чином для DataGrid.

П'ятий рівень містить компоненти, які допомагають користувачам у реалізації власних брокерів ресурсів та планувальників завдань таким чином, що вони можуть перевірити їхні власні стратегії та алгоритми.

Найвищий рівень дозволяє користувачам реалізовувати власні сценарії роботи та конфігурації системи для тестування їх власних політик і алгоритмів.

Функціонування системи.

Процес виконання завдання GridSim представлений на рисунку 1.4. Об'єкт GridUser представляє користувача. Завдання від користувача направляється на відповідний Брокер Ресурсів та у вигляді об'єкта Gridlet. Gridlet є пакетом, який містить всю інформацію, пов'язану із завданням – довжину завдання (виражену в мільйонах операцій, MI), розмір вхідних та вихідних файлів, характеристики користувача. Ці основні параметри

допомагають визначити час виконання, час, необхідний передачі файлів і повернення обробленого Gridlets творцю разом із результатами. Брокер ресурсів запитує у GridInformationService (GIS) список доступних GridResource. GridInformationService реєструє ресурси, що дає змогу відстежувати список доступних грід-ресурсів. Після цього, Брокер ресурсів виконує вибір ресурсу і направляє Опб1е1 на обробку, відповідно до встановленого сценарію.

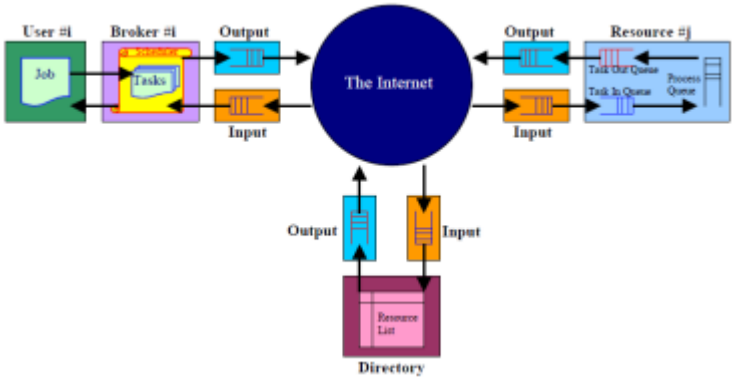

Рисунок 1.4 – Діаграма потоків даних у GridSim [29]

GridResource – об'єкт, складений із однієї чи більше машин. Машина містить один або більше CPU (processing elements, PE), для кожного з яких потужність задається кількістю операцій за секунду (Millions of instructions per second (MIPS)). Після виконання Gridlet, GridResource повідомляє Брокера про завершення завдання. Слід зазначити, що ресурс може виконувати кілька завдань одночасно.

Статистика експерименту.

GridSim дозволяє зібрати статистику за всіма або відібраними операціями, що залежить від того, що конкретно цікавить користувача. Це може бути і моніторинг дискового простору, і розміри черг, і завантаженість мережі та багато іншого. У найпростішому випадку, можна виводити всю інформацію в файл і після моделювання використовувати утиліти роботи з

текстом (наприклад, grep або awk), щоб відібрати дані необхідні для подальшого аналізу. Для графічного представлення результатів можна використовувати Gnuplot.

Копіювання даних.

Починаючи з версії GridSim 4.0, реалізована можливість реплікації даних у грид. Кожен об'єкт Файл має пов'язаний об'єкт FileAttribute, який містить інформацію про розмір файлу, час створення та зміни, майстер-файл або копія і т.д. Replica Catalogue (RC) - відстежує місцезнаходження (копії) файлу. RC забезпечує відповідність між ім'ям файлу та його фізичним розташуванням.

DataGridResource дозволяє користувачам запускати завдання, так і отримувати доступ до даних. DataGridResource передбачає використання таких носіїв даних: HarddriveStorage і TapeStorage. Об'єкт DataGridlet є завданням, для виконання якого потрібно один або більше файлів.

Абстрактний клас ReplicaManager та клас SimpleReplicaManger реалізують основні функціональні можливості копіювання даних. Ці класи відповідають за всі дії з даними DataGridResources, що включає додавання файлу або його копії, реєстрація файлу на RC, видалення файлу або його копії, пересилання потрібного файлу до відповідного DataGridResource.

Візуалізація процесу моделювання та результатів.

Окремим та корисним є питання візуалізації процесу моделювання. Хороший засіб для візуалізації моделювання може заощадити час, витрачений аналіз результатів моделювання. Багато висновків можна зробити лише глянувши на графіки чи діаграми (звісно, за умови що людина, яка аналізує їх, розуміє їхню природу та фізичний зміст).

Починаючи з версії GridSim 5.0 з'явився parallel.gui, в якому міститься інтерфейси Visualizer і AbstractVisualizer, які призначені для візуалізації процесу моделювання. Єдина реалізація цих інтерфейсів, що йде в GridSim – це ParallelVisualizer, який дозволяє з графічного інтерфейсу користувача

запускати, зупиняти експеримент та переходити до режиму покрокового виконання експерименту.

Згідно з офіційною документацією GridSim, ця реалізація повинна використовуватися тільки з метою налагодження, наприклад для покрокової перевірки нових політик або алгоритмів планування. Реальні експерименти не передбачають жодних візуальних інструментів.

Планувальник завдань Alea2.

Робота планувальника Alea2 сконцентрована на моделюванні планування завантаження обчислювального кластера в оточений грід [30]. Alea2 вимагає для роботи GridSim версії не нижче 5.0. Можливості візуалізації у Alea2 набагато перевершують аналогічні у GridSim та PajFit. Він дозволяє спостерігати за зміною стану експериментальної GRID-інфраструктури під час проведення експерименту. Результати виводяться в реальному масштабі часу у вигляді гістограм, графіків та спектральних діаграм. Аналізувати хід виконання експерименту можна за такими критеріями: кількість процесорних елементів: запитаних, використовуваних та доступних; завантаженість кластерів у відсотковому співвідношенні по кожній годині та дню; кількість завдань, які чекають на виконання та виконуються в даний момент часу; середнє використання процесорних елементів кластерів за кожну годину; відсоток відмов ресурсів грід-системи.

Усі результати експериментів, окрім відображення на графіках, зберігаються у текстові файли у форматі CSV. Після проведення експериментів ці дані можна завантажувати у програмні продукти, які дозволяють проводити наступний статистичний аналіз. Наприклад, дані, збережені у форматі CSV, можуть бути легко перетворені на інші формати - XLS (Microsoft Excel), XML, STA (Statistica).

В Alea2 реалізовані алгоритми планування, засновані на чергах, такі як FCFS, EDF, EasyBackfill та ін. Архітектура Alea2 дозволяє проводити поспіль експерименти на одних і тих же вихідних даних (даних про завдання та доступні ресурси) з використанням різних алгоритмів планування. Таким

чином, можна проводити порівняльний аналіз роботи різних алгоритмів планування, їх поведінки у тих чи інших ситуаціях та розробляти сценарії оптимального завантаження наявних ресурсів.

Недоліки бібліотеки GridSim.

В результаті обчислювальних експериментів було виявлено, що обробка запитів на передачу файлів у GridSim недостатньо відображає методи, що використовуються в реальних ситуаціях. Проаналізуємо алгоритм розподілу файлів за структурою грід.

Подія отримання Gridlet DATAGRIDLET_SUBMIT обробляється ReplicaManager, який є інтерфейсом до DataGridlet. Подія обробляється методом receiveDataGridlet(DataGridlet dg), який створює список всіх необхідних файлів. Для кожного файлу перевіряється доступність, якщо файл доступний, він видаляється зі списку. Якщо всі файли доступні (розмір списків нульової), ReplicaManager викликає метод оголошеної політики policy_.gridletSubmit(dg, false), запускаючи таким чином gridlet. Якщо файл недоступний, ReplicaManager надсилає оголошеному в грід ресурсі ReplicaCatalog запит CTLG_GET_REPLICA для визначення ресурсу на якому зберігається файл. ReplicaCatalog відповідає сигналом CTLG_REPLICA_DELIVERY, повідомляючи про положення репліки. ReplicaManager викликає метод receiveReplicaLocation(ev), у якому перевіряє коректність відповіді та посилає на переданий ID ресурсу запит на файл FILE_REQUEST. Цей запит приймає ReplicaManager відповідного ресурсу, який викликає метод processFileRequest(ev), що посилає файл на ресурс, що виконує. У цій останній операції полягає проблема. ReplicaManager, що посилає, використовує метод запису, вказуючи як size довжину файлу. Оскільки операція атомарна, то ніяка наступна передача не може бути здійснена доти, доки не закінчиться попередня. Таким чином, ресурс не може передавати більше одного файлу одночасно. Таке обмеження суттєво видаляє нас від реального життя, до якого модель має бути максимально наближена.

Ця проблема не може бути вирішена багатопотоковою передачею, оскільки java_sim не передбачає таких методів.

Пропозиція по модифікації бібліотеки GridSim.

Для того, щоб обійти дане обмеження та емулювати передачу одночасно кількох файлів, проведено таку модифікацію алгоритму.

На стороні віддає файл ReplicaManager після отримання сигналу FILE_REQUEST викликатиме метод requestFileSRM(ev). Цей метод посилає FILE_REQ_SRM сигнал SRM, який є спадкоємцем sim_java. Отримавши FILE_REQ_SRM SRM підсумовує всі затримки, пов'язані з передачею файлу та його доступністю на дисковому пулі. Якщо потрібно, то SRM емулює завантаження файлу на дисковий пул, тобто. монтування стрічки на драйвер та запис файлу на диск. Коли всі операції закінчуються на бік запиту, сигнал FILE_READY повертається. Після цього ReplicaManager виконує виклик планувальника.

Експерименти, виконані на модифікованій версії алгоритму, показали, що поведінка моделі наблизилася до очікуваного.

Порівняння інструментів моделювання грід систем.

Система моделювання DataGrid повинна мати наступний функціонал і можливості: моделювання основних елементів DataGrid (ресурсів зберігання даних (SE), брокерів ресурсів (RB), каталогу реплік (RC), мережі, користувачів (User), сайтів); швидкість роботи моделі має значно перевищувати швидкість роботи реальної DataGrid; необхідна статистика щодо окремих елементів (наприклад, використання дискових ресурсів) та роботи моделі в цілому (час виконання, кількість переданих файлів, завантаження мережі та ін.); необхідно моделювання збоїв обладнання; результати моделювання мають бути зіставні з реальною ситуацією.

Ми окреслили основні можливості та особливості систем моделювання DataGrid. Порівняємо системи за перерахованими вище вимогами.

Моделювання основних елементів DataGrid.

Користувачі можуть бути змодельовані різними способами. В OptorSim всі користувачі грід представлені одним об'єктом User, який, відповідно до розподілу, створює завдання та спрямовує їх на Брокер ресурсів, який спрямовує завдання на Сайт. У системі Bricks об'єкт Client представляє користувача, який послідовно запускає завдання відповідно до встановленої частоти. Цілком інший підхід реалізований у GridSim, де для користувача можна визначити які завдання він запускатиме і час запуску цих завдань. Останній варіант виглядає найбільш привабливо, але передбачає додаткову доопрацювання програми.

OptorSim не розглядає потреби в обчислювальних ресурсах для виконання завдання. Передбачається, що час виконання завдання складається з часу пошуку потрібного файлу та його копіювання. Bricks та GridSim дозволяють моделювання гібридних ресурсів, які можуть містити обчислювальні елементи та ресурси зберігання даних.

Брокери ресурсів використовуються призначення завдань на ресурси. Для визначення необхідних вирішення завдання ресурсів можуть використовуватися різні алгоритми планування. В OptorSim та Bricks використовується Брокер ресурсів, який просто вибирає найкращий ресурс згідно з обраним алгоритмом. Декілька алгоритмів роботи Брокерів запропоновано в цих системах. У GridSim кожен користувач може мати свій брокер ресурсів, який буде вибирати оптимальні ресурси відповідно до побажань користувача. GridSim не пропонує готових алгоритмів роботи брокерів ресурсів.

OptorSim дозволяє досить спрощено описати мережу – кілька сайтів з'єднуються один з одним. Кожен сайт служить маршрутизатором для пересилання даних сусідам. Bricks дозволяє реалізувати складні топології мереж, але ця процедура є досить трудомісткою. GridSim пропонує інструменти для найбільш повного опису топології ґрід-мережі: зв'язки між елементами, роутери, пакетна передача даних, MTU і т.д.

Статистика.

Статистична інформація, зібрана під час моделювання, необхідна для оцінки якості побудованої моделі та подальшого аналізу результатів моделювання.

Інформація про об'єкти Грид та їх використання у GridSim зберігається у GridStatistics.

Усі результати експериментів, окрім відображення на графіках, зберігаються у текстові файли у форматі CSV. За бажанням можуть бути зібрані дані за всіма параметрами моделі, що цікавлять. Подібна статистика може бути доступна в Bricks. OptorSim надає необхідну статистичну інформацію без додаткового опрацювання програми з боку користувача. Використовуючи графічний інтерфейс, користувач може спостерігати стан моделі навіть у її виконання, а після виконання користувач може отримати повний звіт з графіками, діаграмами і таблицями про результати моделювання.

Моделювання збоїв обладнання.

Моделювання різноманітних збоїв і відмов не представлено ні в Bricks, ні в OptorSim. GridSim дозволяє моделювати відмови ресурсів. Схематично процес виявлення несправності зображено рисунку 1.5.

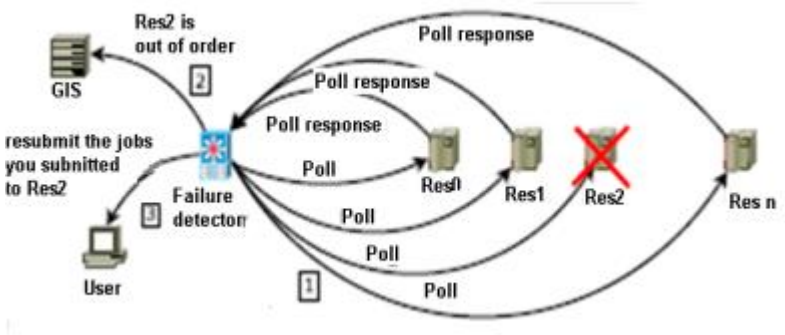

Рисунок 1.5 – Виникнення несправності ресурсу

Детектор помилок регулярно опитує елементи системи. При виникненні помилки інформація про несправний ресурс надсилається до

інформаційного центру, який містить інформацію про стан всіх елементів системи. Запуск завдань на несправному ресурсі буде припинено до відновлення ресурсу. Завдання, які виконувались на ресурсі, у разі несправності будуть перенаправлені на інші ресурси.

Виявлення помилок і відновлення роботи ресурсів невід'ємна частина мереж. Можливість моделювання подібних ситуацій дозволить дослідити їх та покращити роботу реальних грід-систем. Однак відмови ресурсів не єдина проблема БаІаОгій. При передачі даних можуть виникати помилки - по таймуутах, програмні помилки, помилки користувачів та додатків. Логічним шляхом зменшення числа таких помилок є своєчасне реагування на їх появу, підвищення кваліфікації користувачів, постійні експерименти та відстеження змін у глобальній інфраструктурі. Моделювання таких помилок необхідне для адекватного представлення процесів у ВаІаОгій.

Проведений аналіз систем моделювання БаІаОгій показав, що моделювання БаІаОгій складне та трудомістке завдання. Для моделювання складних розподілених систем найбільше підходить ОгійЗіш. Відкритий код та вільне поширення ОгійЗіш також є великим плюсом і передбачає можливість доопрацювання системи новими модулями та алгоритмами. Однак розглянуті системи моделювання не передбачають зміни параметрів потоків даних та завдань на основі даних моніторингу.

2 РОЗРОБКА МОДЕЛІВ, МЕТОДІВ, АЛГОРИТМІВ І ПРОГРАМНИХ ІНСТРУМЕНТІВ ДЛЯ СИНТЕЗУ

2.1 Реалізація інтерфейсу

Насамперед, слід було сформулювати базову концепцію організації програмного інструментарію, щоб він міг виконувати синтез процесів моделювання та моніторингу:

- метою моделювання сучасного обчислювального центру є задоволення деякого критерію оптимальності, що мінімізує вартість обладнання за безумовного виконання SLA (Service Level Agreement);
- найкращий спосіб динамічно оцінити якість роботи системи – використовувати засоби моніторингу;
- програма моделювання має бути поєднана з реальною системою моніторингу розподіленої системи обробки даних через базу даних (БД);
- доцільно прийняти двояку структуру моделі, яка складається з ядра – його стабільної основної частини, незалежної від об'єкта, що моделюється, і декларативного модуля для введення параметрів моделі, що визначають конкретний розподілений обчислювальний центр (його налаштування та параметри, отримані з інформації моніторингу, як потік даних, потік завдань і т.д.);
- БД містить опис інфраструктури, кожного її вузла, зв'язків між ними, інформацію про запущені завдання, часи виконання, результати моніторингу роботи різних підсистем, а також результати моделювання;
- веб-портал необхідний для зв'язку імітаційної моделі та БД, вибору параметрів моделювання та збереження результатів у БД.

Для реалізації системи моделювання потрібно було описати основні об'єкти та події грід та хмарних систем. Також потрібно розробити структуру бази даних, яка міститиме опис 1Т-інфраструктури, кожного її вузла, зв'язків

між ними, дані моніторингу роботи різних підсистем та результати моделювання. Крім цього, необхідно розробити інструментарій, що дозволяє автоматично генерувати вхідні параметри моделі, реалізувати інтерфейси для редагування параметрів моделі системи обробки даних та відображення отриманих результатів.

Програмний інструментарій реалізує ідею синтезу процесів моніторингу та моделювання грід та хмарних систем. На основі даних моніторингу однієї з грід-систем, що зберігаються в базі даних, через веб-інтерфейс виконується їхній статистичний аналіз, результати якого дозволяють потім генерувати потік завдань для зміни параметрів моделювання.

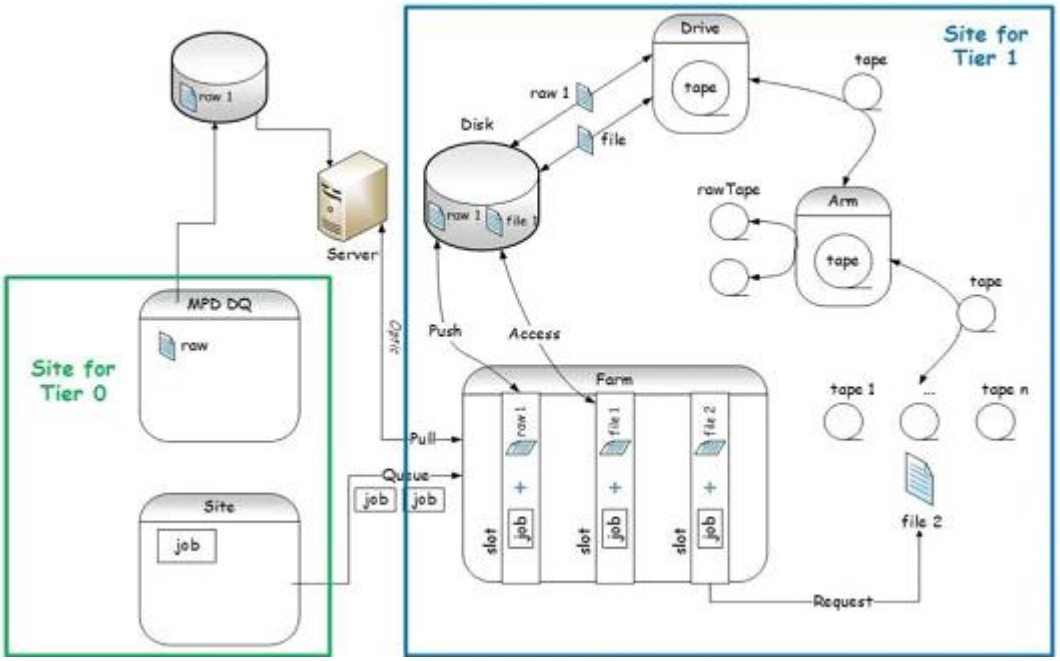

Рисунок 2.1 – Розробка програмного забезпечення. Блоки системи

Інструментарій дозволяє обробку потоку завдань 1Т-інфраструктурою, що володіє заданими ресурсами та правилами їх резервування та використання. Базовий функціонал імітаційної моделі реалізований шляхом доопрацювання та розширення класів системи моделювання Спй81ш. Для

вирішення поставлених завдань розроблено додаткові класи, об'єкти, генератори, інтерфейси та модулі. Об'єкти реалізуються у вигляді java-класів, при цьому користувач задає лише зовнішній опис їх кількості та зв'язків між ними, але не змінює код програми. Події, що відбуваються в моделі, прив'язуються до внутрішнього часу. Результатом роботи моделі є величина часу обробки завдання та розуміння того, як структура обчислювальної установки та продуктивність окремих її частин впливають на цей час. Інше питання, на яке отримують відповідь при моделюванні, які резерви має обчислювальна установка, тобто яка верхня межа інтенсивності потоку завдань (даних) і які компоненти установки на неї істотно впливають.

Засіб опису обчислювальної інфраструктури реалізовано як базу даних із веб-інтерфейсом (рисунок 2.2). Опис присвоюється ідентифікатор, який користувач повинен вказати в параметрах запуску моделі. Модель зчитує інформацію з бази та будує у пам'яті опис обчислювальної структури. Іншими даними є характеристики потоку завдань, які підлягають обробці та інформація. Блок опису потоку завдань - набір утиліт, який дозволяє статистично проаналізувати результати моніторингу та сформувати потік завдань, аналогічний або відрізняється від проаналізованого на керований вплив користувача, у вигляді впорядкованої послідовності записів у базі даних.

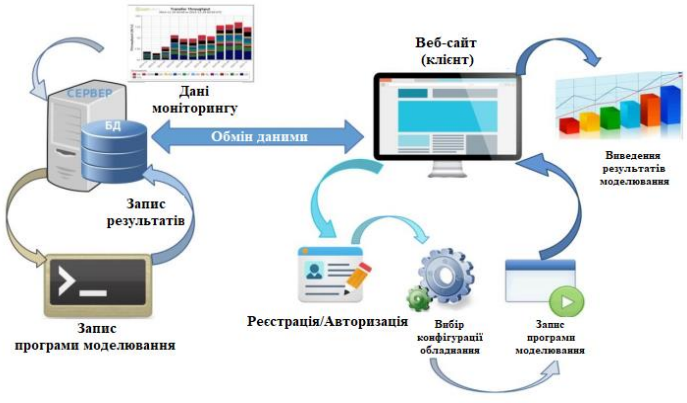

Рисунок 2.2 – Схема роботи розробленого програмного засобу

Ядром моделі є планувальник – модуль, який у програмі відповідає за прийом завдань на обробку та «запуск» їх на процесорах. Планувальник керує чергами, зберігає інформацію про зайнятих та вільних на даний момент процесорів.

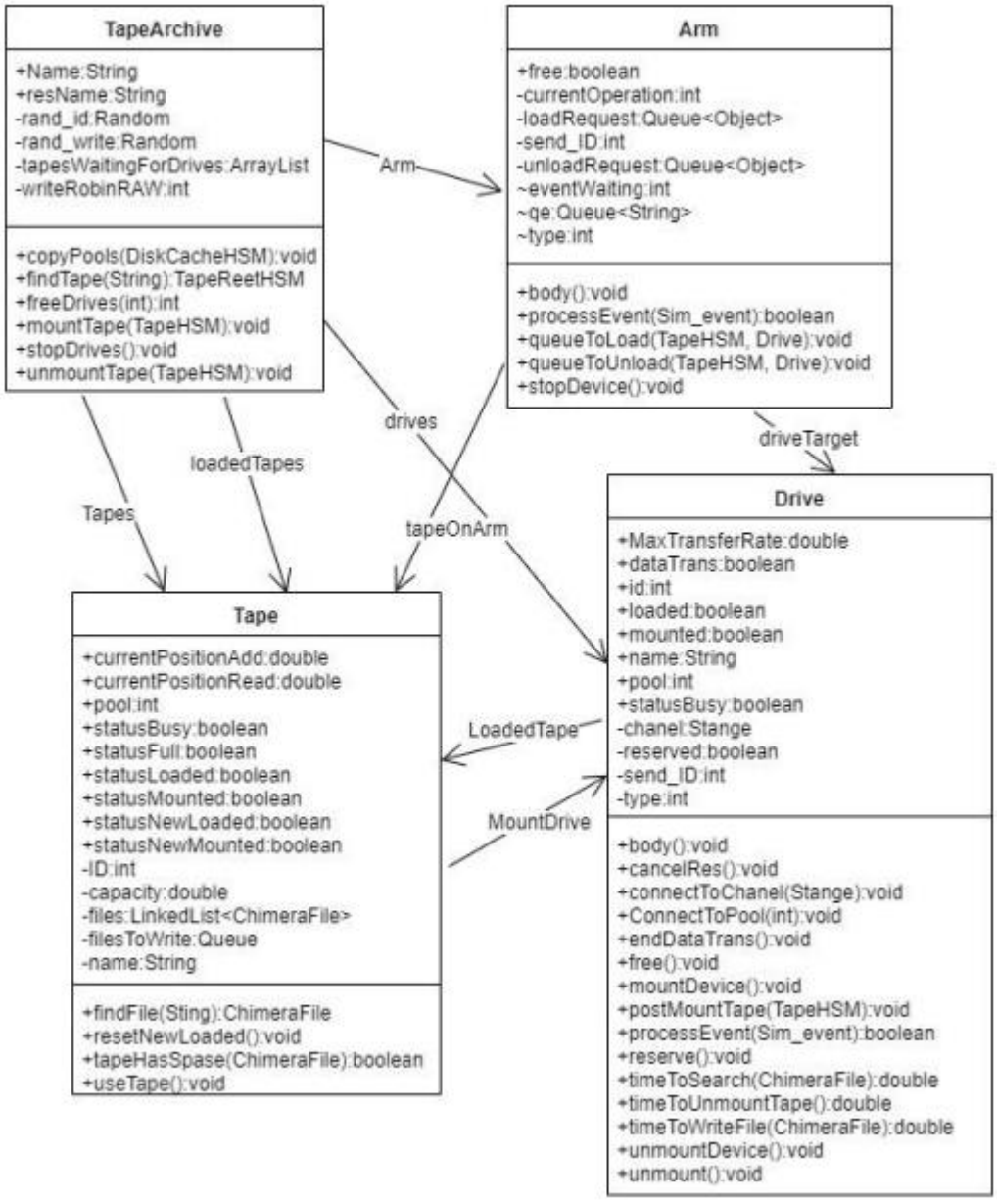

Рисунок 2.3 – Реалізація класів бібліотеки

Інший суттєвий за обсягом набір класів визначає міграцію даних у системі. На відміну від існуючих рішень, де процес передачі даних

реалізований на рівні пакетів, передача даних реалізована через моделювання послідовності подій маніпуляції з файлами запит – відкриття – передача – закриття. У цьому розраховується рівень навантаження середовище передачі, а чи не час надходження пакета на вузол.

Процес імітаційного моделювання полягає у проходженні набору завдань через 1Т-інфраструктуру. Об'єктами моделі є завдання, процесори, файли, стрічки, дискові сховища, лінії передачі, роботизовані бібліотеки. Список подій, що відбуваються з об'єктами, включає: надходження завдання в чергу, вилучення завдання з черги, заняття або звільнення обчислювального ресурсу, передачу файлу, вилучення стрічки зі сховища, монтування, читання – запис на стрічку та ін. Набір реалізованих класів дозволяє імітувати всі процеси , що відбуваються у системі.

Для моделювання роботизованої бібліотеки були розроблені класи, представлені на рисунку 1.3. Набір цих класів дозволяє моделювати процеси, що відбуваються з копією файлу на стрічках: завантаження та вивантаження стрічки маніпулятором, монтування на драйві, пошук файлу на стрічці та його читання/запис.

Процес моделювання інфраструктури з роботизованою бібліотекою полягає в наступному. Файли мають бути записані на стрічки бібліотеки. Паралельно до цього процесу виконуються завдання обробки. Кожне завдання вимагає єдиний файл, копія якого може бути на дисковому масиві, або на стрічці. Завдання починає виконуватися, якщо є вільний слот та всі файли доступні на дисковому сховищі. Якщо файл зберігається в роботизованій бібліотеці, завдання резервує слот, але виконання затримується до його завантаження на диск. Процес переміщення файлу з бібліотеки в дискове сховище включає операцію приміщення стрічкового картриджа (tape) на драйв (drive), яку виконує рука робота (arm), монтування файлової системи картриджа на драйві і запису файлу на диск. Копії файлів на дискових накопичувачах можуть стиратися збирачем сміття, якщо до них довгий час немає звернень. Такий алгоритм роботи з файлами відповідає

алгоритму, реалізованому в системі dCache (www.dcache.org), що має широке поширення в центрах обробки даних.

Вхідний потік завдань для моделювання формується через БД. Для цього реалізована можливість отримання та зберігання даних моніторингу роботи обчислювального центру, щоб використовувати їх як вхідні дані для моделювання. За відсутності необхідної статистики по обчислювальному центру або моделювання нового послідовність виконання обчислювального експерименту буде така: аналізується статистика завдань з аналогічних обчислювальних центрів; будуються розподілу завдань за часом виконання та розміром вхідного файлу; далі висувається та перевіряється гіпотеза про кількість типів завдань у потоці.

Результатом роботи програми моделювання служить послідовність записів у базі даних, що відбиває всі події, які у системі. До них відноситься, наприклад, надходження завдання, початок та кінець обробки завдання, початок та кінець передачі файлу, маніпуляції зі стрічками і т. д. Усі події описуються в єдиному форматі. Запис прив'язується до внутрішнього часу.

Перед проектувальниками обчислювальних установок природно постає питання, як забезпечити масштабованість, тобто здатність установки зберігати працездатність зі збільшенням потоків даних і завдань. У SyMSim налаштування топології установки та характеристик її вузлів конфігурується через зовнішні параметри, що дозволяє легко налаштовувати модель під різні варіанти потоків. Найпростіший параметр при налаштуванні – це кількість рахункових вузлів. Однак стверджувати, що швидкість обробки потоку завдань лінійно залежатиме від нього, невірно. На швидкість впливатимуть також пропускна здатність каналів і продуктивність обладнання, на якому зберігаються вихідні дані. Можливість обліку цих факторів закладена у SyMSim.

При створенні моделі треба також звертати увагу на термін служби обладнання, що передбачається. Наприклад, рахункові вузли можуть замінюватися протягом кількох років, а роботизовані бібліотеки

використовуються кілька п'ятиріч. Отже, на моделюванні процесів у них має бути зроблений особливий акцент.

2.2 Розробка бази даних

Методи, що розробляються в рамках дисертації, повинні вирішувати завдання взаємодії системи моделювання та БД для більш точного прогнозу розвитку системи обробки та зберігання даних великих наукових експериментів.

Існують різні системи управління завданнями та даними, наприклад, PanDA [52] , DIRAC [53] . PanDA — система управління навантаженням у розподіленому гетерогенному обчислювальному середовищі, що реалізує високопотокову концепцію обробки даних. Ця система останнім часом розглядається як перспективна для кількох нових чи модернізованих експериментів.

Тому проектування системи моделювання та пов'язаної БД здійснювалося на підставі результатів аналізу проходження завдань в інфраструктурі під управлінням системи РапОЛ.

Реальна грід система працює наступним чином:

- є сервер, якому з БД вирушають завдання;
- до сервера звертається пілот, повідомляє сайт, на якому він запустився, та запитує завдання;
- якщо завдання для цього сайту є, сервер відправляє його на виконання.

Завдання створити програмне оточення з можливістю збереження даних моніторингу роботи сайту в БД, щоб використовувати її як вхідні дані для моделювання.

Моделювання полягає в наступному:

- на основі статистики генеруються вхідні дані для моделі: тип завдання (симуляція – вимогливий до процесорів, реконструкція –

вимогливий до пам'яті або аналіз), споживання пам'яті, час запуску та завершення задачі, статус завдання, витрачений комп'ютерний час, кількість подій, джерело завдання;

- генерується потік завдань та зберігається в БД;
- модель запитує завдання із БД;
- після виконання завдання аналізується результат роботи, та записується статистика в БД.

Опис бази даних.

Важливою частиною розробки програми було створення БД і програмного оточення взаємодії з нею для збереження даних моніторингу та результатів роботи програми. Результати моделювання записуються у БД. Для реалізації БД було прийнято рішення використати вільно розповсюджувану СУБД PostSql.

Була розроблена структура БД, зображена на рисунку 2.4. БД ключує , наприклад, такі таблиці:

- Users – таблиця, яка містить інформацію про користувача: логін та пароль для входу в систему, e-mail адресу;
- Experiments – таблиця, яка містить інформацію про існуючі експерименти;
- Users_Experiments – таблиця, яка містить інформацію про зв'язок користувачів з експериментами;
- Simulation_Parameters – таблиця, яка містить опис запусків (прогонів) програми моделювання;
- Configurations – таблиця, яка містить опис конфігурації запусків (прогонів) програми моделювання;
- Results – таблиця, яка містить результати запусків (прогонів) програми моделювання;
- Hard_Description – таблиця, яка містить опис параметрів
- процесорів, їх кількості, обсяги дискового сховища тощо;
- Hsm_Description – таблиця, яка містить опис параметрів

- драйвів та стрічок, їх кількості тощо;

- jobswaiting4 – таблиця, яка містить опис потоку завдань (вхідні дані для моделі, на підставі яких генерується потік завдань);

- Communication_Type – таблиця, яка містить опис типів об'єктів (диски, стрічки тощо);

- Communication_Object – таблиця, яка містить опис об'єктів у структурі (диски, стрічки і т.д.);

- Communication_Topo – таблиця, яка містить опис зв'язків між об'єктами у структурі.

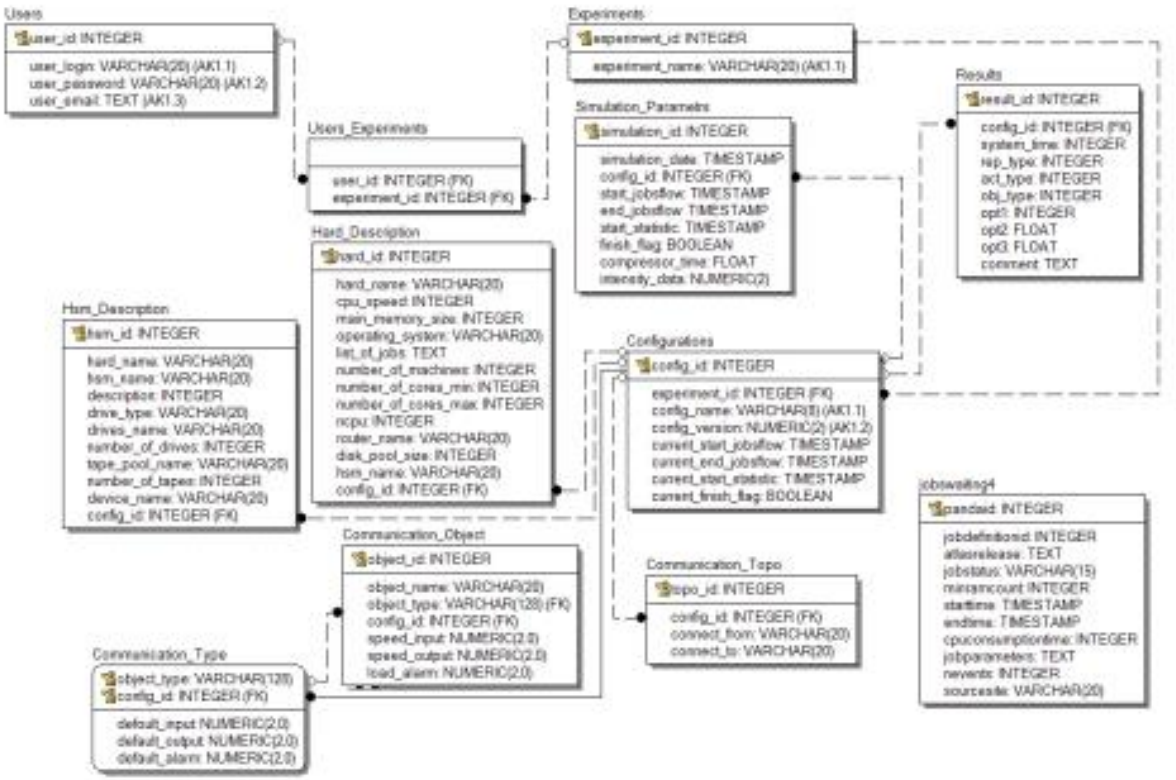

Рисунок 2.5 – Частина структури БД

Випадкові послідовності з різними дискретними та безперервними розподілами, що імітують вищезгадані випадкові величини, а також такі параметри роботи системи як, наприклад, кількість заявок у черзі, час очікування завантаження, тривалість виконання завдання генеруються за

стандартними алгоритмами моделювання випадкових величин з відповідними розподілами з використанням генератора. чисел с рівномірним розподілом в інтервалі (0,1).

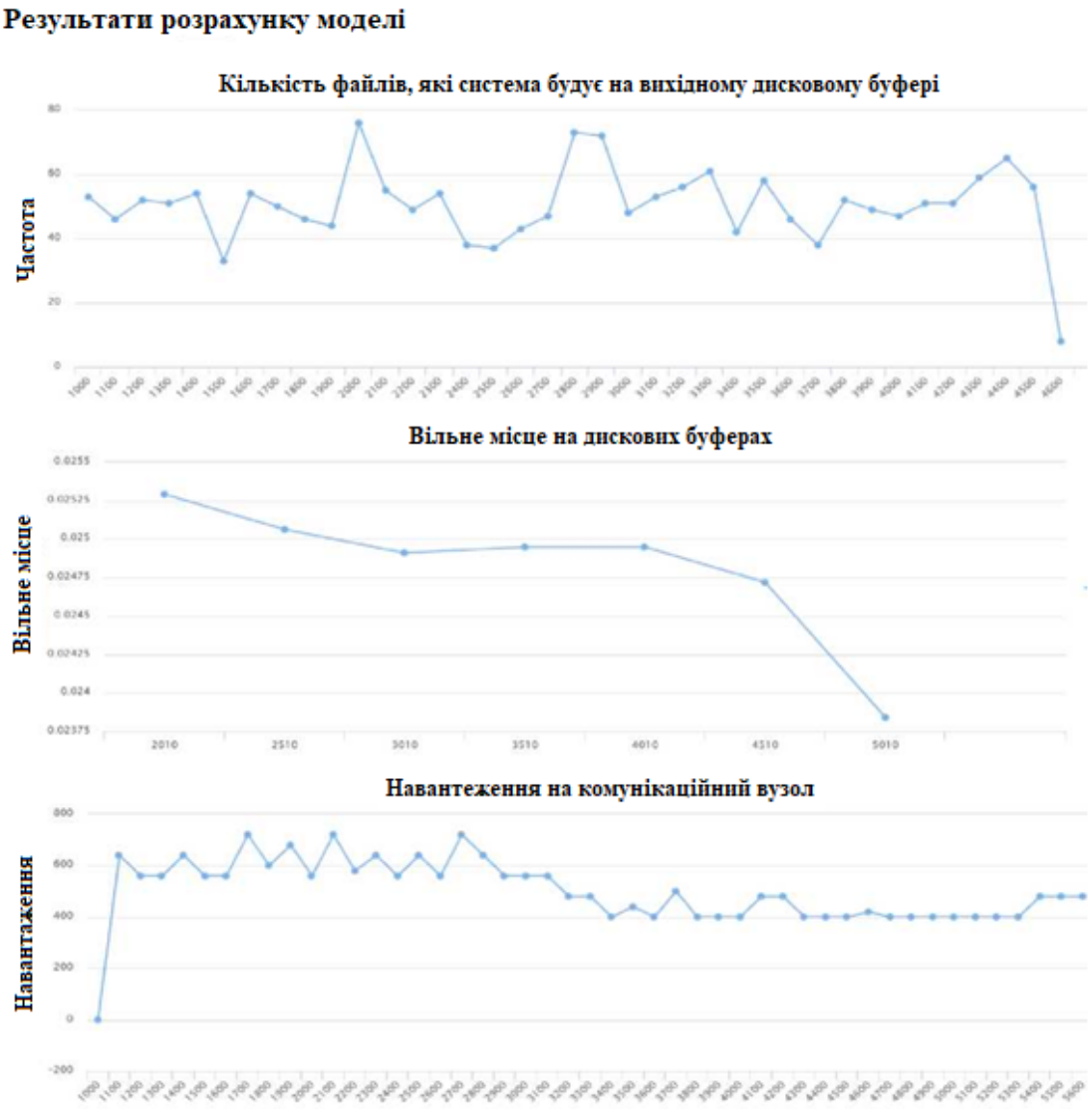

Рисунок 2.6 – Результати моделювання

Завантаження опису моделюється обчислювальної структури реалізовано через веб-інтерфейс. Генерація потоку завдань здійснюється через завдання необхідних параметрів потоку завдань, кожному за періоду випадково визначається кількість завдань, обчислюються випадкові величини: кількість процесорного часу, пам'яті та подій завдання, час старту, і навіть

генеруються додаткові параметри. Модель запускається на сервері. Для цього було написано скрипт, який отримує необхідну інформацію з БД, а потім за допомогою командного рядка запускає модель. Під час роботи моделі на веб-сайті відображається поточний стан моделі, а також кількість записів у результуючій таблиці БД для поточного експерименту. Після завершення моделювання можна переглянути результати. Для цього слід вибрати номер експерименту з випадаючого списку. На сторінці з'являться графіки, що описують різні характеристики системи, що моделюється. Результатом роботи моделі є знайдена величина часу обробки потоку завдань для різних варіантів структури обчислювальної установки та продуктивності окремих її частин, що дозволяє оцінити, як ці фактори впливають на час обробки. Також користувач отримуєдані про навантаження на ресурси системи (CPU, RAM, дисковий буфер), час очікування завдань у черзі, пропускну спроможність мережі, дані щодо навантаження на робота стрічкової бібліотеки. Крім того, в результаті моделювання можна уточнити які резерви має обчислювальна установка, тобто яка верхня межа інтенсивності потоку завдань (даних) і які компоненти установки мають на неї істотний вплив.

ВИСНОВКИ

Проведено аналіз методів та засобів моделювання розподілених систем обробки та зберігання даних. Досліджено підходи до моделювання розподілених систем зберігання та обробки даних великих обсягів. Розроблено підхід для моделювання систем зберігання та обробки даних з використанням результатів моніторингу. Розроблено програмні засобі для моделювання систем зберігання та обробки даних. Результатом роботи моделі є знайдена величина часу обробки потоку завдань для різних варіантів структури обчислювальної установки та продуктивності окремих її частин, що дозволяє оцінити, як ці фактори впливають на час обробки. Також користувач отримує дані про навантаження на ресурси системи (CPU, RAM, дисковий буфер), час очікування завдань у черзі, пропускну спроможність мережі, дані щодо навантаження на робота стрічкової бібліотеки. Крім того в результаті моделювання можна уточнити які резерви має обчислювальна установка.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Білоконь А. С., Борисов С. О., Сальніков С. С., Федорченко В. М. Аналіз функціонування розподілених систем обробки та зберігання даних// Системи управління, навігації та зв'язку. 2024. № 3. С.47-51.

2. Майер-Шенбергер В., Кукьер К. Большие данные. Революция, которая изменит то, как мы живем, работаем и мыслим. // М.: Манн, Иванов и Фербер, 2013. – 240 с.

3. Laney D. Application Delivery Strategies [Электронный ресурс] – Электрон. текст. – 2001. – Режим доступа: https://studylib.net/doc/8647594/3d-datamanagement--controlling-data-volume--velocity--an..., свободный (дата обращения – 12.05.2024).

4. Foster I., Kesselman C. The grid: blueprint for a new computing infrastructure. // Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1999. – 593 p. – ISBN 1-55860-475-8.

5. Попков Ю.С. Макросистемы и grid-технологии: моделирование динамических стохастических сетей // Информационные технологии в управлении. 2003. – N1. – С. 10-20

6. Klusacek D., Matyska L., Rudova H. Alea — Grid scheduling simulation environment // 7th International Conference on Parallel Processing and Applied Mathematics (PPAM 2007), volume 4967 of LNCS. – 2008. – P. 1029-1038.