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 моделювання розподілених систем обробки даних
(тема)

Виконав:

студент II курсу, групи СПм-20-3

Новіков В.С.
(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. Носик А.М.
(посада, прізвище, ініціали)

Допускається до захисту

В.о. зав. кафедри ЕОМ _____ Волк М.О.
(підпис) (прізвище, ініціали)

2022 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 Новікову Владиславу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи моделювання розподілених систем обробки даних

затверджена наказом по університету від " 24 " березня 2022 р. № 413 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 травня 2022 р.

3. Вхідні дані до роботи
розподілені системи
обробка даних
грід система
хмарна інфраструктура

4. Перелік питань, що потрібно опрацювати у роботі
Аналіз предметної області та інструментів моделювання
Системи моделювання хмарних інфраструктур
Розробка системи монгіторингу

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	Аналіз предметної області	28.03.2022–10.04.2022	
2	Дослідження існуючих методів та моделей	11.04.2022–16.04.2022	
3	Розробка методу	17.04.2022–26.04.2022	
4	Проведення моделювання та перевірка роботи	27.04.2022–02.05.2022	
5	Отримання та аналіз результатів	03.05.2022–06.05.2022	
6	Оформлення пояснювальної записки	07.05.2022–13.05.2022	

Дата видачі завдання 28 березня 2022 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Носик А.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 75 с., 11 рис., 1 табл., 1 дод., 7 джерел.

АРХІТЕКТУРА, ГРІД СИСТЕМА, ХМАРА, ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Метою кваліфікаційної роботи є аналіз методів та засобів моделювання розподілених систем зберігання та обробки даних з урахуванням результатів їх моніторингу.

У ході виконання кваліфікаційної роботи проведено дослідження різних підходів до моделювання розподілених систем зберігання та обробки даних великих обсягів; проведено аналіз підходів для моделювання систем зберігання та обробки даних з використанням результатів моніторингу як вхідні параметрів динамічної корекції параметрів моделі; розроблене програмне забезпечення для моделювання систем зберігання та обробки даних, що реалізує ідею синтезу процесів моніторингу та моделювання.

ABSTRACT

Master’s thesis: 75 pages, 11 figures, 1 table, 1 appendices, 7 sources.

ARCHITECTURE, GRID SYSTEM, CLOUD, SIMULATION MODELING, SOFTWARE.

The major goal of this thesis is analyze the methods and tools for modeling distributed data storage and processing systems, taking into account the results of their monitoring. In the course of the qualification work

In order to a study of different approaches to the modeling of distributed storage systems and data processing of large volumes; the analysis of approaches for modeling of systems of storage and data processing with use of results of monitoring as input parameters of dynamic correction of parameters of model is carried out; developed software for modeling storage and data processing systems, which implements the idea of synthesis of monitoring and modeling processes.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

СЕ – обчислювальний елемент

ІТ – інформаційні технології

РЕ – елемент обробки

ЕОМ – електронна обчислювальна машина

ВАК – великий андронний коллайдер

ПЗ – програмне забезпечення

ЦЕРН – Європейська організація з ядерних досліджень (фр. Conseil Europeen pour la Recherche Nucleaire)

ЦПУ – центральний процесор (англ. CPU)

ВСТУП

Сучасні системи зберігання та обробки даних – це складні розподілені програмно-апаратні комплекси, побудовані з 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 приступити до створення розподіленої IT-інфраструктури, необхідно вирішити, яка буде її архітектура залежно від вартісних факторів та інтенсивності потоків даних. З цією метою необхідно виконати моделювання розглядуваної обчислювальної структури, щоб отримати динамічної моделі, що враховує реальну специфіку системи та потоків даних, що надходять, вибрати оптимальну архітектуру.

Як показав проведений аналіз доступних аналітичних методів моделювання, в силу обмежених теоретичних передумов вони не можуть бути застосовані для моделювання складних комп'ютерних комплексів багаторівневої архітектури з реальними розподілами вхідних потоків завдань, складної 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, являє собою логіко-алгоритмічний опис поведінки окремих елементів системи та правил їх взаємодії, що відображають послідовність подій, що виникають у моделі, що моделюється.

Імітаційне моделювання грид 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 в функціонування обчислювального центру, тестувати алгоритми управління завданнями та розподілу ресурсів за групами користувачів. Найчастіше моделювання застосовують лише на етапі проектування грід та хмарних систем, однак, експерименти продовжуються роками і десятиліттями, при цьому обсяги інформації, що обробляється тенденції зростання, тому одночасно з експлуатацією системи відбувається її розвиток, як якісний, а й кількісний. Очевидно, що для досягнення оптимальних результатів моделювання має носити постійний характер на протязі всього життєвого циклу експериментів.

В даний час процеси моделювання та моніторингу розглядаються, як незалежні завдання, які пов'язані між собою. Щоб підвищити точність одержуваних результатів, необхідно як вхідні дані 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 зберігання та обробки даних з урахуванням результатів їх моніторингу.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- дослідження різних підходів до моделювання розподілених систем зберігання та обробки даних великих обсягів;
- аналіз підходів для моделювання систем зберігання та обробки даних з використанням результатів моніторингу як вхідні параметрів динамічної корекції параметрів моделі;
- створення ПЗ для моделювання систем зберігання та обробки даних, що реалізує ідею синтезу процесів моніторингу та моделювання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІНСТРУМЕНТІВ МОДЕЛЮВАННЯ

1.1 Зберігання та обробка Великих Даних

Стрімкий прогрес комп'ютерних технологій, програмних засобів та вибуховий розвиток глобального інформаційного простору, що виник з появою Інтернету, який 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 та мислимо», характеризують Великі Дані, як «масу нових завдань, що стосуються суспільної безпеки, глобальних економічних моделей, недоторканності приватного життя, усталених моральних правил, правових відносин людини, бізнесу та держави» [2].

Раніше збір та обробка великих обсягів даних були недоступні людині, а статистика, теорія ймовірності, дозволяли уточнювати результати обчислень за рахунок отримання великого кількості точок даних. Big Data ж виникає тоді, коли для обробки доступні величезні кількості вимірів. Автори книги приділяють значну увагу розгляду прикладів використання великих обсягів даних, переконуючи у цьому, що ми 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 безлічі інших джерел. Тому часто Big Data визначають, як дані, які занадто великі та складні, щоб їх можна було ефективно запам'ятати, передати та проаналізувати стандартними засобами доступних БД та інших наявних 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).

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) у ЦЕРН [1]. За час першого запуску ВАК у 2012 чотири експериментальні установки на ньому ALICE, ATLAS, CMS та LHCb видавали кожну секунду один петабайт (1015 байт) даних. Запам'ятати таке кількість даних було неможливо на жодній з обчислювальних систем. Тому після надшвидкої складної електронної передобробки, в ЦЕРН виконувалася їх первинна реконструкція в комп'ютерному центрі обробки багато 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, 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 оброблюваних даних про те, 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 відносно короткі терміни. 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 Великих Даних

1.2.1 Грід системи

Термін "грід-обчислення" з'явився на початку 1990 років у "The Grid: Blueprint for a new computing infrastructure» під редакцією Яна Фостера та Карла Кессельмана як метафора, що демонструє можливість простого доступу до обчислювальним ресурсам, як і електричної мережі (power grid). «Грід - географічно розподілена інфраструктура, що поєднує безліч ресурсів різних типів (процесори, довготривала та оперативна пам'ять, сховища та БД, мережі), доступ до яких користувач може отримати з Термін «Грід-обчислення» з'явився на початку 1990 років у «The Grid: Blueprint for a new computing infrastructure» під редакцією Яна Фостера та Карла Кессельмана як метафора, що демонструє можливість простого доступу до обчислювальним ресурсам, як і електричної мережі (power grid).

«Грід - географічно розподілена інфраструктура, що поєднує безліч ресурсів різних типів (процесори, довготривала та оперативна пам'ять, сховища та БД, мережі), доступ до яких користувач може отримати з будь-якої точки, незалежно від місця їх розташування. Грід припускає колективний режим доступу до ресурсів і до пов'язаних з ними послуг у рамках глобально розподілених віртуальних організацій, що складаються з підприємств та окремих фахівців, спільно які використовують загальні

ресурси.

У кожній віртуальній організації є своя власна політика поведінки її учасників, які мають дотримуватись встановлених правил. Віртуальна організація може утворюватися динамічно та мати обмежений час існування» [7]. Автори у статті «Грід-технології: статус та перспективи» перераховує основні програми грид, до яких відносяться:

- 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 напрямків 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, в 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 є створення в ЦЕРН БАК. Дуже великий обсяг даних, що надходять із детектора, вимагає використання грід системи для їх обробки. Ця система має кілька рівнів організації та є розподіленою моделлю зберігання та обробки даних. Базовими елементами є комп'ютерні вузли Tier0, Tier1, Tier2. Суть розподіленої моделі у тому, що весь обсяг інформації з детекторів ВАК після

обробки в реальному часі та первинної реконструкції повинен спрямовуватися для подальшої обробки та аналізу у регіональні центри.

Центром обробки даних є Tier0, розташований у ЦЕРН. Він здійснює первинну обробку даних, їх калібрування, створює файли. інформації для подій Після первинної обробки ці файли розсилаються у 13 обчислювальних центрів Tier1 з достатньою ємністю зберігання та цілодобовою підтримкою грид, де проводиться основний масив обчислень з реконструкції даних вимірів. Ці результати у формі файлів з реконструйованими даними розсилаються до центрів рівня Tier2, де стають доступними фізикам, які виконують їхній аналіз. Для індивідуальної роботи або роботи невеликої групи фізиків з певній темі призначені локальні обчислювальні бази Tier3, функції якої вже може виконувати окремий персональний комп'ютер.

1.2.2 Хмарні обчислення

«Хмарні обчислення – це модель надання зручного мережевого доступу в режимі «на вимогу» до набору, що колективно використовується настроюваних обчислювальних ресурсів (наприклад, мереж, серверів, сховищ даних, додатків та сервісів), які користувач може оперативно задіяти під свої завдання та звільняти при зведенні до мінімуму кількості взаємодій із постачальником послуги або власних управлінські зусилля. Ця модель спрямована на підвищення доступності обчислювальних ресурсів». Споживач, коли йому необхідно, може самостійно задіяти обчислювальні можливості, такі як серверний час або мережеве сховище даних, в автоматичному режимі, без взаємодій з персоналом постачальника послуг. Всі можливості доступні через мережу на основі стандартних механізмів, що забезпечує використання різнорідних тонких та товстих клієнтських платформ, таких як мобільні телефони, ноутбуки. Постачальник об'єднує свої обчислювальні ресурси в пул обслуговування великої кількості споживачів. Різні фізичні та віртуальні ресурси динамічно розподіляються і

перерозподіляються в відповідно до потреб користувачів. Виникає відчуття незалежності від розташування, коли замовник не знає та не контролює, де безпосередньо знаходяться обчислювальні ресурси, якими користується. Обчислювальні можливості можуть швидко та гнучко резервуватися (часто автоматично) для оперативного масштабування під завдання замовника, а також швидко звільнятися. З погляду споживача доступні можливості часто виглядають нічим не обмеженими та можуть бути придбані у будь-якій кількості у будь-який час. Хмарні системи автоматично контролюють та оптимізують використання ресурсів через вимір деяких параметрів, таких як розмір сховища даних, обчислювальна потужність, пропускна здатність, кількість активних користувацьких записів. Таким чином, хмарні технології забезпечують мережевий доступ до обчислювальним, програмним та інформаційним ресурсам (мережам передачі даних, серверів, пристроїв зберігання, сервісів та додатків), конфігурується відповідно до оперативних запитів. Вони дозволяють значно скоротити витрати на IT-інфраструктуру, задовольняти динамічно мінливі потреби у ресурсах тощо.

1.2.3 Гібрідні розподілені системи обробки даних

Для широкого спектра завдань у різних галузях науки є актуальним скорочення часу їх виконання, а також підвищення ефективності використання ресурсів. Актуальною є розробка методів створення багатофункціональних гетерогенних комплексів для вирішення широкого класу задач 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, що вимагало значного збільшення обчислювальних потужностей та ресурсів зберігання даних.

Слід зазначити, що в прошлому році не відбувся третій запуск ВАК після його суттєвої модернізації, де очікувалось дворазове збільшення даних, що збираються, і неминучий перехід до гібридних IT-інфраструктур. Запуск колайдера відкладено через пандемію коронавірусної інфекції та пов'язані з цим затримки з підготовкою до роботи головних детекторів CMS та ATLAS. Це необхідно для потенційно нової фізики, але стикається з новими серйозними вимогами, а саме:

- значним збільшенням обчислювальних потужностей та мережевих ресурсів зберігання даних;
- необхідністю доступу до даних із грід та хмар;
- активному використанню розподілених паралельних обчислень;
- вдосконаленню кодів програм аналізу та моделювання.

На сьогоднішній день грид 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 вимагає 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 метод наукового пізнання, то моделювання це один з методів пізнання, що полягає в описі реальних об'єктів та явищ за допомогою інших об'єктів та 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, відмінною від реального втілення.

За способами моделювання розрізняють:

- фізичне, натурне моделювання;
- структурно-функціональне (за допомогою мови, блок-схеми, графіка, картки);
- математичне (за допомогою аналітичного опису: рівнянь та формул).

Оскільки імітаційне моделювання виникло і може бути виконано лише на комп'ютері, його часто називають комп'ютерним моделюванням, що взагалі кажучи, звужує опис цього методу моделювання, т.к. в даний час на комп'ютері реалізуються і аналітичні та різні числові моделі явищ.

Як цілі моделювання, можна назвати: осмислення, вивчення насправді; спілкування; навчання, тренаж. Функції моделювання: конструювання та перевірка моделі; експериментування з моделлю для вивчення поставленого завдання, вибору оптимальної стратегії її вирішення або передбачення поведінки досліджуваного процесу у майбутньому.

Моделі класифікують:

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

Якщо тепер вибирати спосіб моделювання, найбільш адекватний Вирішуване завдання, то очевидно, що фізичне моделювання нам не підходить. При фізичному моделюванні досліджувана система замінюється відповідною їй іншою матеріальною системою, яка відтворює властивості досліджуваної системи із збереженням їхньої 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 для різних варіантів. Перевірка на на практиці близько десятка різних типів умов пов'язана з великими зусиллями, тимчасовими та чималими матеріальними витратами.

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 виявляється 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 можуть бути формули або рівняння, набори правил або угод, виражені в математичної форми.

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 досліджуваної системи шляхом послідовного виконання великої кількості елементарних операцій. Аналітичне моделювання дозволяє отримати більш точне рішення, оскільки головним завданням є вирішення рівнянь для отримання теоретичних результатів та їх зіставлення із практикою. Але за допомогою цього виду моделювання дуже складно провести повне дослідження процесу функціонування складної системи, вивчити її загальні властивості вивчати складні системи стало можливим у процесі розвитку інформаційних технологій, коли комп'ютери стали використовувати для моделювання

процесів функціонування системи 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 назва імітаційного моделювання. Такі моделі є комп'ютерну програму, яка крок за кроком відтворює події, що відбуваються у реальній системі.

Перевагою імітаційних моделей є можливість підміни процесу зміни подій у досліджуваній системі в реальному масштабі часу на прискорений процес зміни подій темп роботи програми. Говорячи про те, яку технологію застосувати для моделювання грид і хмарних систем, необхідно враховувати, що можливість застосування аналітичних моделей для розглянутих 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 дисципліни обслуговування, викликаними теоретичними причинами;
- система розглядається як динамічна стохастична мережа, описується системами рівнянь, що дозволяють враховувати, як маршрутизацію, і розподіл ресурсів у мережі, причому вивченню підлягають рівноважні та нерівноважні стани мережі.

Обидва підходи видають результат моделювання, як правило, у вигляді асимптотичних розподілів та в силу обмежених теоретичних передумов 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 ресурсів. 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 і процес її розвитку в часі має бути описаний.

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

Технології комп'ютерного моделювання широко використовуються в теперішній 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).

Комп'ютерне моделювання є одним із ефективних методів вивчення складних систем, так як комп'ютерні моделі простіше та зручніше досліджувати в силу їх можливості проводити обчислювальні експерименти, які в порівнянні з реальним експериментом утруднені через фінансових та фізичних перешкод або можуть дати непередбачуваний результат. Логічність та формалізованість комп'ютерних моделей дозволяє виявити основні фактори, що визначають властивості досліджуваного об'єкта оригіналу (або цілого класу об'єктів), зокрема, досліджувати відгук модельованої фізичної системи на зміни її параметрів та 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. Це напрямок отримав назву імітаційного моделювання. Такі моделі є комп'ютерною програмою, яка крок за кроком відтворює події, що відбуваються у 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.

Для запуску програми моделювання грид-структурипотрібно задати склад і топологію центрів обробки моделюваної грид структури, а також розподіл ресурсів між завданнями. Після цього програма виконує імітаційне моделювання процесів проходження згенерованого набору завдань через цю грід структуру. В якості результатів обчислюються часові оцінки шуканих параметрів потоку завдань. Моделювання системи дозволяє відповісти на низку питань щодо підбору кращих властивостей системи. При створенні розподіленої системи потрібно прийняти рішення щодо архітектури інфраструктури, кількості ресурсних центрів, обсягу потрібних 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 одночасно з експлуатацією відбувається не лише якісне, але й та кількісний розвиток системи. Таким чином, навіть за 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.

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. В якості даних для такого прогнозу можна використовувати статистику, накопичену в час роботи програми моніторингу моделі, що моделюється. При цьому необхідно врахувати, що модель має розглядатися як невід'ємна частина системи обробки даних, а дані моніторингу як вхідні моделювання. Це дозволить приймати більш обґрунтовані проектні рішення у розвитку системи. Також об'єднавши моделювання та моніторинг в рамках одного програмного пакета, можна досягти суттєвого зниження експлуатаційних витрат та вкладень у збільшення потужності з метою збереження швидкості отримання результату експериментів, постійне підвищення потоку даних.

1.4 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 грид. 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 тощо; при відключенні частини обчислювальних ресурсів або додавання нових ресурсів; при збільшенні кількості даних, що

передаються; при виході з ладу частини комунікаційних каналів При цьому оцінка ефективності управління може проводитись за наступним найбільш популярним критеріям:

- 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, якщо використати ефективні методи моделювання.

Грід-системи, орієнтовані на зберігання та обробку великих масивів інформації називаються DataGrid. DataGrid технологія високо затребувана у наукових колах: астрономія, моделювання білка, фізика високих енергій. Експерименти у цих областях генерують велике кількість даних, які мають бути загальнодоступними для вчених для обробки та аналізу. У пакетах моделювання Brick, OptorSim та GridSim зроблено спробу моделювання DataGrid систем, що може послужити основою подальших розробок у цій галузі.

1.4.1 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, що запускають мережі завдання на загальні ресурси. Програма була реалізована на Java.

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 користувальницьку машину, яка генерує завдання виконання на ресурсі. Об'єкт Server характеризується продуктивністю, робочим навантаженням та їх 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. Завдання за замовчуванням виконується однією 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 QueueTimeSharing. Топологія мережі повинна бути визначена користувачем конфігураційний файл.

Bricks включає в себе засоби моніторингу, прогнозування та 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, приймаючи його увагу інформацію про стан ресурсу, отриману з ResourceDB та модуля прогнозування.

Функціонування системи.

Перед запуском моделі користувач має визначити частоту надходження завдань від клієнта та обсяги завдань. Робота системи починається з аналізу конфігураційних файлів Коли нове завдання надходить від Клієнта, воно обробляється Планувальником, відповідальним за розміщення завдання з

урахуванням її особливостей. Планувальник запитує інформацію у модулі Прогнозування щодо можливості використання того чи іншого ресурсу та розміщує завдання на відповідний Сервер.

Сервер обробляє завдання з черги. Коли сервер завершив обробку завдання, результати пересилається Клієнту. Статистика експерименту та 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 інше.

Копіювання даних Структура Bricks дозволяє моделювати множинне копіювання даних (реплікації). Реплікація даних та автоматичний розподіл задач по ресурсах гарантує баланс навантаження та полегшує доступ до даних всім користувачам.

Охоплення всіх часових поясів також полегшує цілодобовий моніторинг та підтримку.

Копіювання даних може здійснюватися згідно з централізованою чи ієрархічної моделі. У централізованій моделі всі дані та всі завдання, що мають справу з цими даними, зберігаються та обробляються на єдиному сайті, у якого, як передбачається, є достатні ресурси для обробки та відповідна ємність запам'ятовуючих пристроїв.

Ієрархічна модель передбачає обробку завдань, коли зі збільшенням завантаження сервера, частина даних копіюється на ресурси нижнього рівня. ReplicaManager виконує алгоритм копіювання файлів та збору інформації про дискові ресурси.

Щоб уникнути нестачі дискового простору, Bricks стежить за вільним простором на дискових ресурсах та видаляє дані, коли це необхідно.

Критерієм видалення даних є відношення використаного дискового простору до всього обсягу. Цей поріг визначає користувач конфігураційному файлі. Коли граничне значення перевищено, відбувається видалення файлів.

1.4.2 Система моделювання OptorSim

OptorSim створювався у рамках європейського проекту European DataGrid (EDG) як інструмент для моделювання DataGrid. У додатках, працюючих з великими обсягами даних, важливо не лише уникнути втрати даних, а також організувати оптимальний доступ до них. OptorSim – пакет моделювання DataGrid, реалізований мовою Java, дозволяє оцінювати різні алгоритми оптимізації та стратегії копіювання. Вихідний код широко доступний.

В OptorSim кожен сайт може містити кілька елементів зберігання (StorageElement (SE)) та/або обчислювальних елементів (ComputingElement (CE). У пакеті кілька конфігураційних файлів використовуються для завдання параметрів моделі. Детальний опис файлів та характеристик представлено у посібнику користувача. Їх створення та доопрацювання моделі не викликає особливих труднощів. Існує можливість моделювання трафіку фону. Відкритий доступ до вихідного коду OptorSim дає можливість доопрацювання системи самостійно. Архітектура Система складається з двох частин: перша призначена для моделювання грид-ресурсів, а друга частина – для моделювання запуску завдань користувачів.

Мережа складається із Сайтів (GridSites), пов'язаних між собою. Топологія мережі визначається користувачем у конфігураційному файлі. Основною ідеєю є те, що кожна задача складається з множини файлів даних (набір даних пов'язаних із цим завданням). Перш ніж копіювати файл на ресурс визначається "найкраще" розташування копії файлу, що дозволяє Сайтам копіювати файл з різних джерел, щоб уникнути надмірної завантаженості ресурсів. Передбачається, що для виконання завдання немає необхідності в обчислювальних ресурсів. StorageElements дозволяє виконати різні маніпуляції з файлами: додавання та отримання доступу до файлу, видалення файлу. В разі, коли необхідно видалити файли, можливе використання різних сценаріїв видалення файлів. Наприклад, можна

видаляти найменш використовувані файли, або найстаріші файли. ReplicaCatalogue (RC) – список відповідності логічних імен файлів їх фізичних імен. ReplicaManager (RM) керує копіюванням даних, керує файлами та реєструє їх у ReplicaCatalogue.

Пересилання даних між сайтами здійснюється RM та контролюється ReplicaOptimiser (RO). RO містить алгоритми копіювання даних, які забезпечують обмін даних, створення та видалення копій. Мета моделювання - досягти оптимізації роботи мережі за рахунок об'єднання зусиль з оптимізації окремих RO на веб-сайтах. ResourceBroker (RB) вирішує який Сайт надати для виконання завдання. В OptorSim реалізовано кілька алгоритмів роботи брокера: RandomCEResourceBroker посилає GridJob випадковому CE; AccessCostResourceBroker вибирає CE, якому знадобиться найменший час для отримання всіх необхідних файлів; QueueLengthResourceBrokers вибирає CE з найменшою чергою; CombinedCostResourceBroker: для виконання завдання підбирає CE, якому знадобиться найменший час для отримання всіх необхідних файлів у всіх завданнях, що перебувають у черзі на цьому CE.

Моделювання проекту GridPP та випробувального стенду CMS DataChallange показали, що при використанні алгоритму RandomCEResourceBroker час виконання завдання найбільший, алгоритм QueueLengthResourceBroker виконує ті самі завдання приблизно вдвічі швидше. AccessCostResourceBroker вимагає менше часу, але характеризується низьким використанням CE, оскільки вибираються CE з великою кількістю зв'язків. До того ж, сценарії RandomCEResourceBroker та QueueLengthResourceBroker швидко призводять до заповнення дискового простору SE (до 90%), у той час як AccessCostResourceBroker використовує SE приблизно 37%. Найкраще використання CE, хороші показники використання SE (приблизно 75%) та найменший час виконання задач показав алгоритм CombinedCostResourceBroker, тому що оцінює і комунікативні можливості Сайту, та довжину черг завдань, що дозволяє

досягти найкращого балансу між плануванням завдань на найближчі даних Сайти та завантаженістю Сайтів. Варто відзначити, що вибір відповідного алгоритму роботи брокера вказується у конфігураційному файл.

Поведінка Користувача (User) визначається у конфігураційному файлі і відрізняється розподілом чи частотою, з якою вони запускають завдання на RB. Так, за сценарієм SimpleUsers, користувачі запускають завдання згідно рівномірному розподілу (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, який містить інформацію про ім'я файлу, розмір, чи є файл копією або майстер файлом. Майстер-файл - оригінальна копія даних, яку не можна видалити. Набір данних (Dataset) – кілька файлів, які мають спільні властивості. Кожна Завдання може містити один або більше наборів даних. Параметри доступу (access patterns) визначають порядок обробки файлів у задачі.

Функціонування системи. На рисунку 1.1 показано алгоритм роботи ResourceBroker. Користувач запускає нові завдання, які RB обробляє по черзі. Завдання має бути надіслано для виконання на сайт, у якого є доступ до всіх файлів у завданні. Коли RB знаходить невиконану роботу, виконується пошук Сайту з відповідним CE. Якщо знайдений CE є вільним, завдання буде спрямована на Сайт та додана у чергу на обробку на CE. Далі CE моделює запуск завдання. Для кожного файлу в задачі визначається “найкраще” розташування файлу для копіювання, згідно алгоритм 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 елементів36 копіюють файли по одній мережі, то пропускна здатність ділитися між CE. Таким чином, кожен наступний файл повинен чекати, поки

попередній файл буде передано.

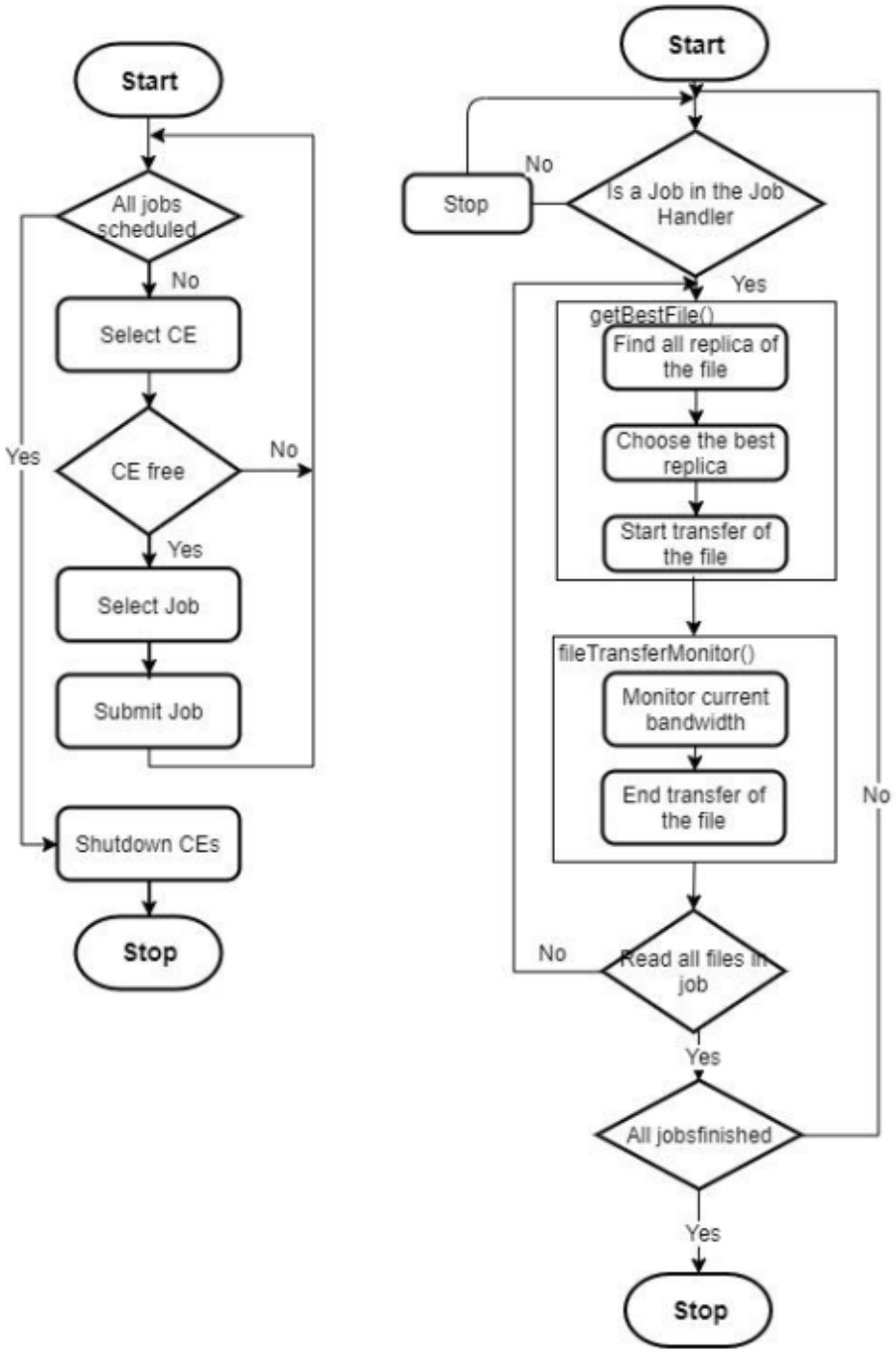

Рисунок 1.1 – Алгоритми роботи RB та CE

Статистика в OptorSim представлена як по кожному елементу окремо, так 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 процентне співвідношення часу використання CE на час простою, кількість віддалених і 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. Інтерфейс 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 в табличній формі. Закладка Logical View показує граф, на якому відображаються поточні передачі файлів між сайтами. Докладна інформація представлена у посібнику користувача.v

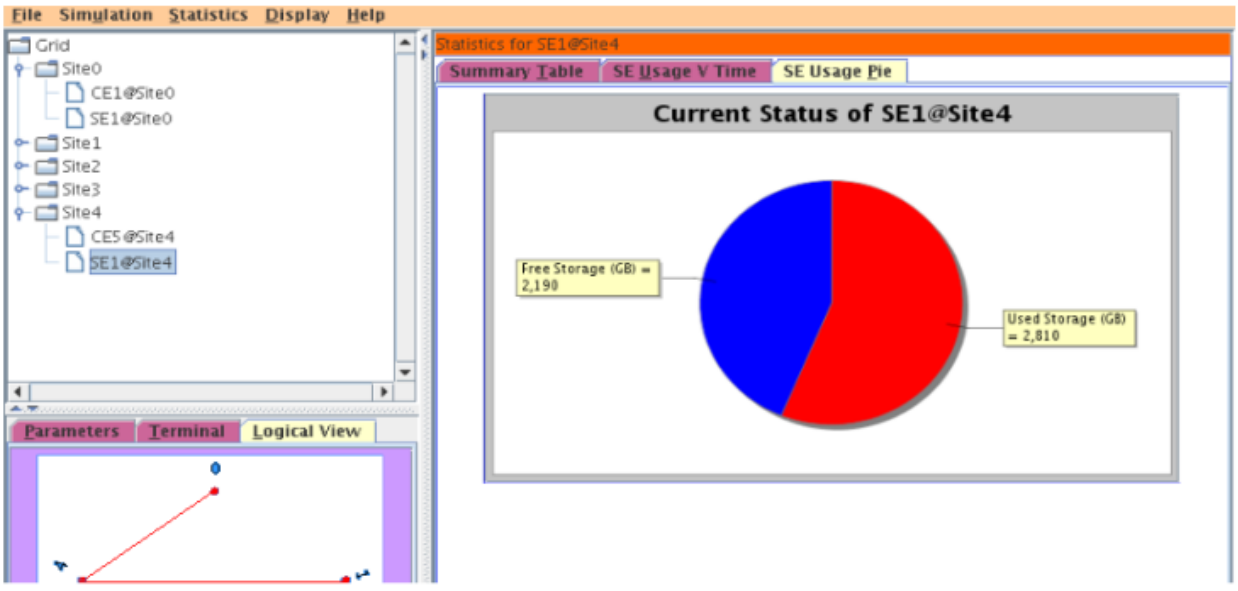

Рисунок 1.2 – Графічний інтерфейс OptorSim

1.4.3 Система моделювання GridSim

Проект GridSim розробляється групою дослідників у лабораторії з вивчення хмарних та розподілених обчислень відділу інформатики та комп'ютерних обчислень в Мельбурнський університет, 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 задач на обчислювальних кластерів. За допомогою GridSim можна проводити експерименти, що відтворюються, які складно реалізувати в теперішньому оточенні динамічних грід-систем.

Основні можливості GridSim полягають у наступному:

а) моделювання різних характеристик ресурсів грід-середовища;

б) моделювання різних політик планування завдань на вузлах обчислювальних кластерів - як вже реалізованих (FCFS, Easy Backfill, Conservative Backfill), і розроблених користувачами алгоритмів;

в) використання даних про завантаження реальних кластерів для проведення експериментів;

д) підтримка механізму аукціону для планування завдань;

е) моделювання різних конфігурацій обчислювальної мережі грідсистеми для різних топологій;

ж) моделювання регіональних компонентів грід, інформаційних сервісів.

GridSim проектувався як багаторівнева система моделювання, можливості якої можуть бути легко та значно розширені. Архітектура GridSim представлена рисунку 1.3. GridSim заснований на SimJava - бібліотеці для створення дискретно-подійних процесів, реалізованому мовою

Java. Тому SimJava займається обробкою внутрішніх подій та взаємодії компонентів GridSim. Усе компоненти GridSim взаємодіють між собою через відправлення повідомлень, визначених у SimJava.

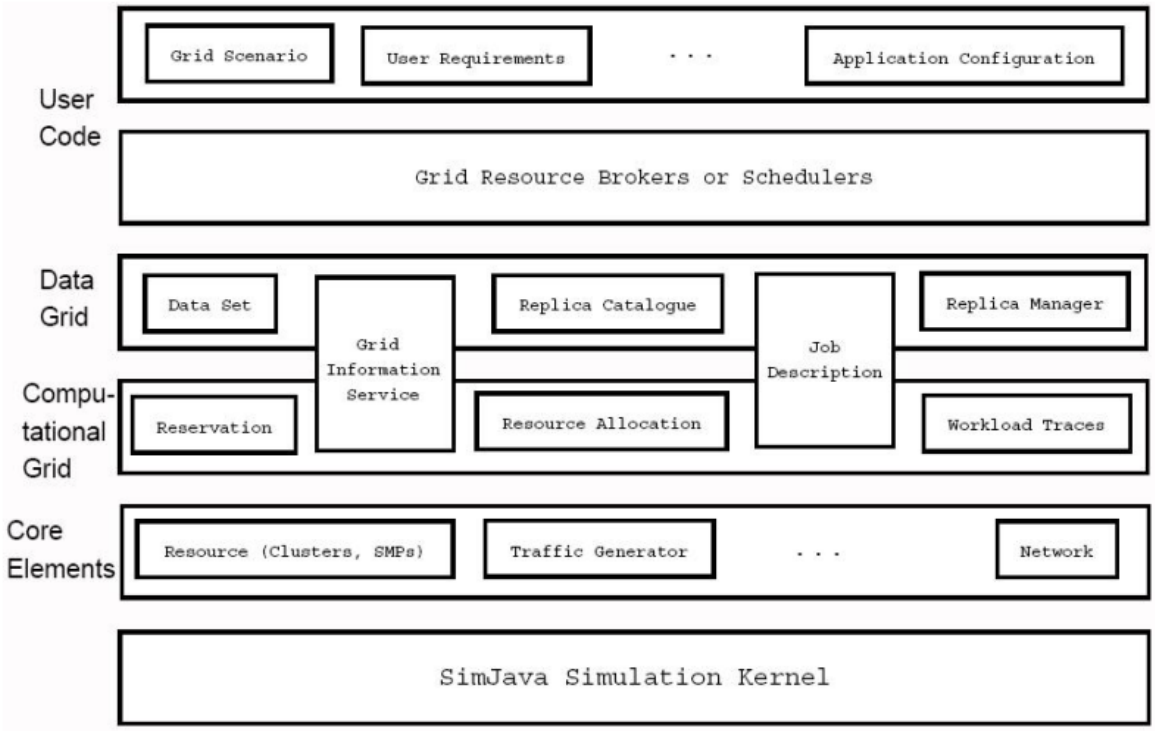

Рисунок 1.3 – Архітектура GridSim

Другий шар моделює основні компоненти розподіленої інфраструктури, а саме, грид-ресурси, наприклад, кластери, сховища даних та мережеві з'єднання. Ці компоненти істотні 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 та обчислювальними вузлами. Служби управління 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, вони можуть перевірити свої власні стратегії та алгоритми. Найвищий рівень дозволяє користувачам реалізовувати їх власні сценарії роботи та конфігурації системи для тестування їх власних політик та алгоритмів. Функціонування системи Процес виконання завдання GridSim представлений рисунку 1.4. Об'єкт GridUser представляє користувача. Завдання від користувача спрямовується 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 для передачі файлів і повернення обробленого Gridlets творцю разом із результатами. Брокер ресурсів запитує у GridInformationService (GIS) список доступних GridResource. GridInformationService реєструє ресурси, що дає можливість відстежувати список доступних грід-ресурсів. Після чого, Брокер ресурсів виконує вибір ресурсу та спрямовує Gridlet на обробку, згідно з встановленим сценарієм

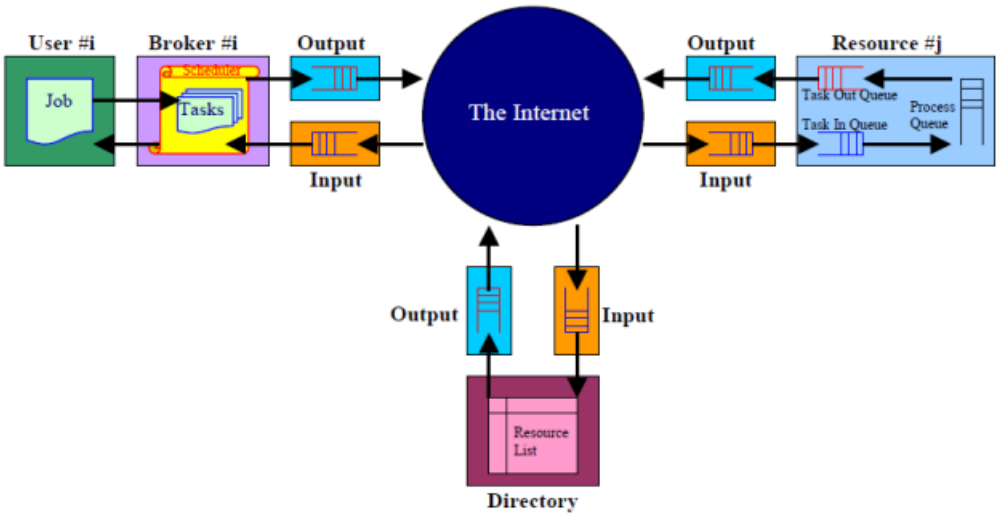

Рисунок 1.4 – Діаграма потоків даних в GridSim

GridResource – об'єкт, який складається з однієї або більше машин. Авто містить один або більше CPU (processing elements, PE), для кожного з яких потужність задається кількістю операцій на секунду (millions of instructions per second (MIPS). Після виконання Gridlet, GridResource повідомляє Брокер про завершення завдання. Слід зазначити, що ресурс може виконувати кілька завдань одночасно. Статистика експерименту GridSim дозволяє зібрати статистику за всіма або відібраними операціям, що залежить від того, що цікавить користувача. Це може бути і моніторинг дискового простору, і розміри черг, і завантаженість мережі та багато іншого. У найпростішому випадку можна виводити всю цікаву інформацію у файл і після моделювання використовувати утиліти роботи з текстом (наприклад, grep або awk), щоб відібрати дані необхідні подальшого аналізу. Для графічного уявлення результатів можна використовувати Gnuplot.

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 чи копія тощо.

Replica Catalogue (RC) – відстежує розташування (копії) файлу. RC забезпечує відповідність між ім'ям файлу та його фізичним розташуванням. DataGridResource дозволяє користувачам як запускати завдання, так і отримувати доступ до даних. DataGridResource передбачає використання наступних носіїв даних: HarddriveStorage та TapeStorage. Об'єкт DataGridlet є завданням, для виконання якого потрібен один або більше файлів. Абстрактний клас ReplicaManager та клас SimpleReplicaManger реалізують основні функціональні можливості копіювання даних.

Ці класи відповідають за всі дії з даними на 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 висновки можна зробити лише глянувши на графіки або діаграми (звичайно, за умови що людина, яка аналізує їх, розуміє їхню природу і фізичний зміст). Починаючи з версії GridSim 5.0 з'явився parallel.gui, в якому міститься інтерфейси Visualizer та AbstractVisualizer, які призначені для візуалізації процесу моделювання. Єдина реалізація цих інтерфейсів, що йде в GridSim - це ParallelVisualizer, який дозволяє із графічного інтерфейсу користувача запускати, зупиняти експеримент та переходити в режим покрокового виконання експерименту.

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

Планувальник завдань Alea2.

Робота планувальника Alea2 сконцентрована на моделюванні планування завантаження обчислювального кластера в грід оточенні. Alea2 вимагає роботи GridSim версії не нижче 5.0. Можливості візуалізації у Alea2 набагато перевершують аналогічні у GridSim та PajFit. Він дозволяє спостерігати зміну стану експериментальної GRID-інфраструктури у час проведення експерименту Результати виводяться у 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 кожному години та дня; кількість завдань, які чекають на виконання та виконуються в даний час; середнє використання процесорних елементів кластерів за кожну годину; відсоток відмов ресурсів грід-системи. Усі результати експериментів, крім відображення на графіках, зберігаються у текстові файли у форматі CSV.

Після проведення експериментів ці дані можна завантажувати в

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, що ґрунтуються на чергах, такі як FCFS, EDF, EasyBackfill та ін. Архітектура Alea2 дозволяє44 проводити поспіль експерименти на тих самих вихідних даних (даних про надходження завдань і доступні ресурси) з використанням різних алгоритми планування.

Таким чином, можна проводити порівняльний аналіз роботи різних алгоритмів планування, їх поведінки в 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 у реальних ситуаціях.

1.5 Порівняння інструментів моделювання

Система моделювання DataGrid повинна мати наступний функціоналом та можливостями: моделюванням 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 по роботі моделі в цілому (час виконання, кількість переданих файлів, завантаження мережі та ін); необхідно моделювання збоїв обладнання; результати моделювання мають бути можна порівняти з реальною ситуацією.

Порівняємо системи за перерахованими вище вимогами. Моделювання основних елементів DataGrid Користувачі (users) можуть бути змодельовані у різний спосіб. 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. Зовсім 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, додаткове доопрацювання програми.

OptorSim не розглядає потреби в обчислювальних ресурсах для виконання завдання. Передбачається, що час виконання завдання складається з часу пошуку відповідного файлу та його копіювання. Bricks та GridSim дозволяють моделювання гібридних ресурсів, які можуть містити та обчислювальні елементи та ресурси зберігання даних. Брокери ресурсів використовуються призначення завдань на ресурси. Для визначення необхідних для вирішення задач ресурсів можуть використовуватися різні алгоритми планування.

В OptorSim та Bricks використовується брокер ресурсів, який просто вибирає найкращий ресурс згідно з обраним алгоритму. Кілька алгоритмів роботи Брокеров 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 є маршрутизатором для пересилання даних сусідам. Bricks дозволяє реалізувати складні топології мереж, але ця процедура досить трудомістка. GridSim пропонує інструменти для найбільш повного опису топології грід-мережі: між елементами, роутери, пакетні передачі даних, MTU і т.і.

2 СИСТЕМИ МОДЕЛЮВАННЯ ХМАРНИХ ІНФРАСТРУКТУР

На сьогоднішній день існує різні системи моделювання хмарних інфраструктур, наприклад: CloudSim, iCanCloud, CReST. Ці програмні продукти дозволяють створювати моделі хмарних систем певною функціональністю та конфігурацією. Готова модель запускається на моделювання, внаслідок чого, системи моделювання надають статистичну інформацію щодо найважливіших характеристикам: час виконання завдань, життєвий цикл віртуальних машин (ВМ); використання ресурсів. Аналізуючи цю інформацію, розробник може виявити вузькі місця у моделі та передбачити їх вирішення, реалізувавши яке можна перевірити наступною ітерацією моделювання.

2.1 Система CloudSim

Система моделювання CloudSim є розширюваним набір інструментальних засобів, що включає моделювання систем хмарних обчислень та оточення, що надається. Інструментарій CloudSim підтримує як моделювання поведінки компонентів хмарної системи, таких як дата-центри, віртуальні машини, так і політику надання ресурсів. Він реалізує універсальні методики надання, які можуть бути розширені з невеликими зусиллями.

В даний час, він підтримує моделювання середовища хмарних обчислень, що входить до складу однієї або декількох хмар (федерація хмар). CloudSim реалізована мовою Java, з використанням системи профайлінгу JProfiler, що сприятливо позначається на зменшенні ризику витоків пам'яті та підвищення стабільності програми при запуску кількох потоків одночасно. Проект активно розвивається, що добре видно по оновленням системи у громадському репозиторії.

Для роботи необхідно завантажити вихідні коди системи CloudSim і створити проект у будь-якій зручній інтегрованого середовища розробки (IDE). Створення моделі проводиться через створення відповідних сутностей у коді та заповненні їх параметрів. Моделювання створеної моделі запускається шляхом компілювання, складання і запуск безпосередньо головного файлу в проекті. Архітектура CloudSim Спочатку CloudSim був заснований на дискретному ядрі моделювання подій SimJava, який успішно застосований у платформі моделювання DataGrid - GridSim.

Однак, ядро SimJava було замінено на інше ядро CloudSim Core Simulation Framework, для забезпечення таких функціональних можливостей як:

- скидання моделювання програмно в режимі реального часу;
- деактивація (утримання) об'єктів;
- перемикання контексту об'єктів між різними станами (наприклад, очікування активності);
- підтримка створення нових сутностей у режимі реального часу та включення їх у модель;
- зменшення витрат продуктивності при моделюванні системи великого розміру.

Шар Simulation CloudSim надає підтримку моделювання віртуального середовища дата-центру, заснованого на хмарі, що включає спеціалізовані інтерфейси управління віртуальними машинами, пам'яттю, сховищами та смугою пропускання. Основні проблеми, такі як забезпечення ВМ хостами, управління виконанням додатків, та моніторинг динамічного стану системи, що контролюються на цьому шарі.

Провайдер хмари, який хоче вивчити ефективність різних політик розподіл його хостів для ВМ, повинен реалізовувати його стратегії на цьому рівні. Таке використання може бути зроблено програмно, розширюючи ключову функціональність надання ВМ. У цьому шарі є чітка відмінність, пов'язане із забезпеченням ВМ хостами. Хмарний хост може одночасно

розташовувати набір ВМ, які виконують програми, засновані на SaaS, надаючи певний рівень QoS. Також цей шар надає функціональність, яку розробники хмарних програм можуть розширювати для виконання складних робочих навантажень та дослідження продуктивність програми. Найвищий рівень у схемі архітектури CloudSim це код користувача, який надає базові сутності для хостів (число машин, їх специфікація і т.д.), додатки (кількість завдань та їх вимоги), ВМ, кількість користувачів та типи їх додатків, та брокер планування політик.

Розширюючи ці базові сутності, розробник програми хмари може здійснювати такі дії:

- генерація поєднань, необхідних розподілених робочих навантажень, конфігурацію додатків;
- для моделі хмари доступні сценарії та навантажувальні тести на основі конфігурації користувача;
- реалізація власних методологій надання додатків для хмар та його федерації.

Функціональність CloudSim. CloudSim дозволяє моделювати:

- основні рівні хмари (IaaS, PaaS, SaaS);
- розподіл ВМ у межах одного або кількох обчислювальних вузлів;
- латентність повідомлень;
- федерацію хмар;
- динамічне робоче навантаження.
- споживання енергії дата-центру;
- динамічне створення сутності моделі.

Модель хмарної інфраструктури у CloudSim наближена до реальної хмари структури. Основні послуги рівня інфраструктури (IaaS) моделюються з урахуванням дата-центру. Дата-центр може керувати декількома хостами, які по черзі керують ВМ під час їх життєвий цикл. Під хостом тут розуміється фізичний обчислювальний сервер у хмарі, якому визначено такі властивості як: обчислювальна потужність (виражена в MIPS), обсяг

оперативної пам'яті, обсяг постійного сховища.

CloudSim дозволяє моделювати політики розподілу ВМ на рівні хоста та завдань на рівні ВМ. Тут політика ВМ означає політику управління операціями, пов'язані з життєвим циклом ВМ, такими як: постачання хостів ВМ, створення ВМ, знищення ВМ та міграції ВМ. Так користувач може визначити чи ВМ працюватимуть паралельно на одному хосте (розподіл ВМ за часом) або послідовно у порядку черги.

Аналогічним чином визначається процес виконання завдань лише на рівні ВМ. Завдання виконуються паралельно на однієї ВМ з урахуванням розподілу по ядрах процесора ВМ або послідовно один за одним. Моделювання в CloudSim базується на подіях, де різні сутності зв'язуються між собою за допомогою передачі повідомлень при виникнення події. Кожне повідомлення затримується при передачі від сутності до сутності такі затримки в мережі моделюються через матрицю затримок. Елементи цієї матриці є час у мілісекундах, яке має пройти при передачі повідомлення від однієї сутності до іншої. Опис топології збережено у форматі BRITE який містить набір мережевих вузлів, які можуть бути більшими, ніж набір модельованих вузлів. Ці вузли представляють різні сутності CloudSim, включаючи хости, дата-центри, хмарні брокери та ін.

В результаті моделювання CloudSim надає докладну статистичну інформацію щодо життєвого циклу ВМ, завантаженості ВМ, черги відправлених, прийнятих та виконаних завдань за часом. При моделюванні федерації хмар, виводиться інформація про завантаженість приватної та публічної хмар. Додаючи в модель характеристики споживання енергії кожною одиницею хмарної інфраструктури, можна отримати дані щодо ефективності споживання електроенергії. Додаткову інформацію завжди можна отримати, додавши розвантаження даних в балку в сутності CloudSim, що цікавлять. У результаті отримані даних достатньо для аналізу роботи спроектованої моделі та політик надання.

2.2 Система iCanCloud

iCanCloud - це ще одна програма для моделювання хмарних інфраструктур. Основною метою iCanCloud є знаходження компромісного рішення між вартістю та продуктивністю для заданого набору додатків, що виконуються на конкретному апаратному забезпеченні, та надання користувачам корисну інформацію про такі видатках. Крім цього, iCanCloud може бути використаний широким колом користувачів, починаючи від основних користувачів, що переходять на хмарну інфраструктуру до розробників великих розподілених додатків.

iCanCloud має графічний інтерфейс користувача, який дозволяє швидко і легко моделювати хмарну інфраструктуру з доступними ресурсами, такими як ВМ, хмарний гіпервізор, програми та апаратну частину. Створення користувальницьких ресурсів здійснюється додаванням власних класів до системи iCanCloud, написаних мовою програмування C ++. Архітектура iCanCloud iCanCloud розроблена як компонент дискретного середовища моделювання подій OMNeT++. Її основна сфера застосування є моделювання мереж зв'язку. Мережа тут позначена у більш широкому значенні і представляє собою провідні та бездротові мережі зв'язку, мережі на чіпі, мережі масового обслуговування тощо.

Для моделювання мереж в iCanCloud використовується INET, який також є компонентами OMNeT++.

Архітектура iCanCloud була розроблена на базовій ідеї систем хмарних обчислень: надання користувачам псевдонастроювану апаратне середовище, де вони можуть запускати свої програми. Нижній рівень архітектури iCanCloud складається із моделей апаратної частини. На цьому рівні моделюються апаратні частини системи: жорсткі диски, модулі пам'яті та центральні процесори. У свою чергу, цей рівень поділений на чотири групи, кожна з яких відповідає певної частини системи: підсистема обробки (CPU), підсистема оперативної пам'яті, підсистема зберігання даних та підсистема

мережі. Базовий системний модуль API безпосередньо з'єднаний із моделями рівня апаратної частини.

В основному цей модуль містить набір системних викликів, які представлені як API (інтерфейс прикладного програмування), для всіх програм, що виконуються у ВМ за допомогою iCanCloud. Таким55 Таким чином, ці системні виклики забезпечують інтерфейс між програмами та сервісами, що надаються моделями апаратної частини. Крім того, дослідники можуть писати програми для моделювання в iCanCloud, використовуючи API. З метою підтримки сумісності, API розроблено за стандартів POSIX.

На рівні вище знаходиться сховище ВМ. Це сховище містить у колекцію ВМ, яку визначає користувач. Спочатку, симулятор надає кілька моделей існуючих ВМ у хмарах Amazon (EC2). Користувач може додавати, видаляти та редагувати ВМ у цьому сховищі.

Кожна ВМ моделюється шляхом налаштування кожної відповідної нижчестоящої моделі апаратної частини. Наступний рівень містить у собі сховище додатків. Аналогічно сховищу ВМ, до цього репозиторію додано набір заздалегідь певних моделей додатків. Ці моделі будуть використовуватися для того, щоб налаштувати відповідні завдання, які будуть виконані в певному примірнику ВМ. Крім того, користувачі можуть легко додавати власні моделі додатків за допомогою API. Рівень вище називається хмарним гіпервізором.

Цей рівень складається з модуля, що відповідає за управління всіх завдань і за створення ВМ до виконання цих завдань. Після того, як робота над завданням закінчена, цей модуль зупиняє ту ВМ, на якій була виконана робота з задачі, і повторно розподіляє ресурси системи для виконання решти задач. Цей модуль також містить політики витрат для призначення вхідним завданням на конкретних примірниках ВМ, розраховані на відповідних евристичні правила. На верхньому рівні розташовується модуль хмарної системи. Цей модуль містить визначення всієї сутності хмарної системи, яка в здебільшого складається з визначення гіпервізора, визначення кожної ВМ,

яка наповнює цю систему.

Функціональність iCanCloud.

Виходячи з архітектури iCanCloud система має наступне функціоналом:

- моделювання існуючих та неіснуючих хмарних систем;
- проста інтеграція та тестування нових та готових політик брокера ресурсів, шляхом гнучкого налаштування модуля гіпервізора;
- налаштування ВМ може бути використане для швидкого моделювання одноядерних та багатоядерних систем;
- налаштування широкого набору конфігурацій системи зберігання даних дозволяє моделювати локальні системи зберігання, видалені системи. зберігання, такі як NFS та паралельні системи зберігання даних, такі RAID та паралельні файлові системи;
- графічний інтерфейс користувача надає просту генерацію та налаштування великих розподілених моделей. Цей графічний інтерфейс націлений насамперед на управління та вивчення раніше налаштованих ВМ, хмарних систем, гіпервізорів, додатків та для отримання графічного звіту;
- POSIX сумісний API та адаптована бібліотека MPI дозволяє моделювати роботу програм.

Також використовуючи функціональність моделювання додатків в iCanCloud, програми можна реалізовувати та моделювати на основі статичних графів, використовуючи трасування реальних додатків та програмування нових додатків безпосередньо у платформі моделювання. У базовому комплекті системи моделювання iCanCloud реалізовані деякі сутності для швидкого створення моделі хмарної інфраструктури.

Створення моделі може бути реалізовано із застосуванням готових сутностей як знизу-вгору за рівнями архітектури, і зверху вниз. Таким чином, iCanCloud дозволяє оперувати готовими хмарами, на приклад приватної хмари, розподілених віртуальних машин, а також відомого хмарного сервісу Amazon_EC2 (посилання на сервіс можна вставити). Результати

моделювання в iCanCloud Завдяки графічному інтерфейсу, в результаті моделювання в iCanCloud користувач отримує наочну структуру спроектованої хмарної системи.

Використовуючи компонент INET для моделювання мережі фізичних та віртуальних обчислювальних вузлів, користувач отримує структури мережі. Після моделювання хмарної системи користувачеві доступні статистичні дані як у графічному, так і у файлі формату *.csv. В останньому відображені точки часу виконання програм, звільнення ресурсів та життєвий цикл ВМ. Інформація у такому вигляді дозволяє створювати необхідні графіки та діаграми для подальшого аналізу моделі.

2.3 Система CReST

CReST - інструментарій, розроблений для моделювання хмарного забезпечення, на основі дискретно-подійного моделювання. Причиною створення CReST послужила необхідність використання робастної системи моделювання для дослідження та вивчення методик управління датацентрами та надання хмарних сервісів.

Інструментарій CReST прагне вийти на функціональність, яка дозволяє в короткі терміни мінімальними трудовитратами виявити проблеми хмарної інфраструктури та вказати на можливі помилки у політиках надання та управління датацентрами.

CReST є автономною системою, написаною мовою програмування Java і є вільно поширюваною. На відміну від інших існуючих систем моделювання хмарних інфраструктур, CReST дозволяє запускати моделювання кількох абстрактних рівнів, починаючи від фізичного рівня, енергоспоживанням та тепловими потоками в межах дата-центру, до мережної інфраструктури та рівня віртуалізації хмарних сервісів, що виконуються на запит користувача.

Архітектура CReST. Система CReST спроектована як набір пов'язаних

модулів, які можуть бути незалежно відключені або включені залежно від вимог до моделі. Вхідними даними для запуску CReST є конфігураційний файл формату XML. Цей файл містить повну специфікацію апаратної частини кожного датацентру. CReST Builder – це графічне додаток для створення та редагування таких конфігураційних файлів. Також система може читати необов'язковий файл "Parameters", в якому користувач вказує вручну параметри, що його цікавлять. Ті у свою чергу перезаписують відповідні параметри у XML-файлі.

Додатково користувачі можуть задавати свої власні події через текстовий файл «User Events», в яких визначено тип події та час події. CReST має графічний інтерфейс. Під час моделювання у режимі реального часу користувач може стежити за збоями, навантаженням серверів, теплового потоку на карті дата-центру через графічний інтерфейс. Для забезпечення розширюваності та модульної незалежності CReST реалізована на базі патерну проектування Model-View-Controller.

Кожен незалежний модуль має абстрактний метод ModuleRunner, який переглядає чергу подій через інтерфейс Java Observer-Observable. Модулі, виймаючи події з черги подій EventQueue, ігнорують їх або приймають на виконання, що в свою чергу може призвести до генерування нових подій та розміщення їх у чергу подій.

Таким чином, модулі - це незалежні спостерігачі черги подій, які взаємодіють між собою через події. Такий підхід дозволяє включати та вимикати модулі, додавати модулі та швидко їх реєструвати в системі.

Моделювання починається створенням World object, який містить у хоча б один або кілька об'єктів дата-центрів. Кожен дата-центр у свою чергу містить перелік абстрактних блокових об'єктів. Блоки реалізовані у чотирьох конкретних типах: Aisle, Container, AirCon та Rack.

Aisle та Container, містить у собі список об'єктів Rack. У свою чергу Rack містить перелік об'єктів Server. Server та AirCon реалізують інтерфейс збоїв Failable. Сервер містить жорсткий диск, оперативну пам'ять, програмне

забезпечення та хоча б один центральний процесор. Server може запускати сервіси (Service) та віртуальні машини (VirtualMachine), які запускаються та зупиняються через методи start() та stop(). Усі оновлення статусу у World object ініціалізуються об'єктами Event створені кожним ModelRunner. Події лягають у чергу подій EventQueue і сортуються за часом.

Коли подія витягується з черги, воно виконується і має можливість створювати нові події, які також лягають у чергу. Як тільки подія отримує статуси «виконаний» або "згенерований", воно проглядається кожним об'єктом ModuleRunner. Якщо ModuleRunner зацікавлений у події, він виконує відповідна дія, інакше подія ігнорується.

2.4 Система моделювання SimGrid

SimGrid є набір інструментів, який забезпечує основні функціональні можливості для моделювання розподілених додатків у хмарних інфраструктурах. Використовуючи фреймворк SimGrid для моделювання, користувачі отримують можливість запускати сотні тисяч ВМ та керувати цими ВМ так само, як і в реальному світі (наприклад, призупиняти/відновлювати роботу ВМ, а також переміщувати ВМ між модельованими серверами).

Користувачі можуть виконувати обчислювальні та комунікаційні завдання на фізичних (PMs) та віртуальних (VMs) машини моделі через SimGrid API, який забезпечує безперешкодний та плавний перехід до моделювання IaaS для сотень користувачів SimGrid. Крім того, програмна реалізація ВМ SimGrid включає модель алгоритм динамічної міграції - precopy. Ця модель коректна розраховує час міграції, а також трафік, створюваний міграцією з враховуючи конфлікт ресурсів, викликаний іншими обчисленнями, що проводяться в ході роботи моделі та обміну даними в рамках усієї моделі для імітації роботи реальних фізичних машин.

Все це дозволяє користувачам отримувати точні результати

моделювання динамічних віртуалізованих систем Точність, як функціонування програмної реалізації віртуальних, що симулюються машин, так і міграції даних машин всередині моделі була експериментально підтверджена розробниками, шляхом проведення мікро-тестування в різних умовах. Також за допомогою спеціально розробленого алгоритму консолідації віртуальних машин SimGrid досягається дуже високий рівень масштабування моделей із значним числом віртуальних та фізичних машин.

Користувачі можуть динамічно створювати завдання CPU-обчислень та мережевих комунікацій. З перебігом внутрішнього часу моделі дані завдання виконуються, споживаючи мережеві та обчислювальні ресурси моделі. Всередині фреймворка SimGrid формалізує завдання задоволення обмежень, виділення частки ресурсів кожному за завдання. Доки не закінчиться моделювання, SimGrid формалізує завдання задоволення обмежень, вирішуючи її на цьому кроці, а потім переходячи до наступного кроку моделювання.

Як вхідні параметри для моделі використовуються XML файли, звані файлами платформи, які описують, як з апаратної точки зору організована модель системи, наприклад, продуктивність процесора, кількість ядер, мережева топологія, сховище.

Сама ж модель описується програмою мовами Java, C, Lua або Рубі. Моделювання у фреймворку SimGrid мовою Java реалізуються при допомоги об'єктів-контейнерів для віртуалізованих навантажень, що є об'єктом класу Process. Об'єкт класу Process ініціалізується параметрами vm та load. Після ініціалізації об'єкти класу Process зв'язуються з об'єктами класу VM (параметрами host, name, nbCores, що ініціалізуються, ramsize, netBW, diskPath, diskSize, migNetBW, dpIntensity), що імітують віртуальні машини. Пов'язані таким чином об'єкти навантаження та об'єкти моделі взаємодіють усередині з інтерфейсом Msg, що забезпечує формалізації даних моделі та передачу формалізованої інформації всередину фреймворку для виконання обчислень на модулі розв'язання систем лінійних рівнянь LMM.

2.5 Порівняльний аналіз систем моделювання

Для проведення порівняльного аналізу систем моделювання хмарних інфраструктур необхідно визначити низку критеріїв. Проведений аналіз дозволив сформулювати наступний набір критеріїв:

- моделювання апаратної частини (рівень PaaS). Завдання та облік технічних характеристик обчислювальних вузлів, аналіз енергоспоживання, аналіз тепловиділення, моделювання збоїв обчислювальних вузлів;
- моделювання рівня віртуалізації (рівень IaaS). Завдання та облік технічних характеристик ВМ, моделювання політик надання ресурсів, моделювання політик розподілу завдань з ВМ, міграція ВМ у межах хмари;
- моделювання поведінки програмного забезпечення у хмарі (рівень SaaS). Завдання та облік складності завдання, моделювання реальних додатків;
- моделювання мережі~~. Моделювання~~ (топології мережі, ~~моделювання~~ затримок між об'єктами, ~~моделювання~~ різних протоколів передачі даних);
- моделювання федерації хмар (приватні, громадські, гібридні);
- наявність графічного інтерфейсу;
- технічна підтримка, відкритий код.

Основною функціональністю системи CloudSim, що виділяє її серед інших, є: моделювання власних політик розподілу ВМ по дата-центру та задач по ВМ, моделювання федерації хмар та зв'язки між приватними та публічними хмарами, моделювання динамічного навантаження з урахуванням відмови та збоїв в апаратній частині датацентру.

Також, варто зазначити, наявність великої кількості реалізованих проектів, які дозволяють провести моделювання області, що цікавить хмарної інфраструктури. До недоліків системи належать: відсутність графічного інтерфейсу та, як наслідок, знання мови програмування Java як інструмент для моделювання, а також складність оцінки трудомісткості реальних програм у форматі MIPS.

Потужним засобом моделювання роботи програм в iCanCloud є базовий системний модуль із набором системних викликів, розроблених за стандартом POSIX. Це дозволяє впроваджувати у модель хмарної64 інфраструктури існуючі програми та аналізувати їх роботу в хмари. Такий підхід знижує накладні витрати на підготовку завдань. моделювання, і, як наслідок, зменшує час моделювання загалом.

Графічний інтерфейс зменшує складність процесу моделювання та запуск на моделі, а також полегшує аналіз результатів. Основною перевагою iCanCloud перед представленими системами моделювання хмарних інфраструктур є наявність модуля цінової політики, яка дозволяє встановлювати ціну на використання ресурсів хмари та проводити їх облік. Однак, iCanCloud не має такого ж широкого набору функціональності для моделювання готових політик надання ВМ та розподілу завдань за ними, як для протоколів передачі даних або системи зберігання.

В таблиці 1.1 представлений порівняльний аналіз систем моделювання.

Таблиця 1.1 – Порівняльний аналіз систем моделювання

Функція	GridSim	SimGrid	OptorSim	iCanCloud	CloudSim
Реплікація даних	+	–	+	–	+
Планувальник завдань	+	+	–	+	+
Генерація фонового мережевого трафіку	+	+	+	+	+
Резервування CPU	+	–	–	+	+
Графічний інтерфейс	–	–	+	–	+
Моделювання гібридних архітектур	–	–	–	–	–
Використання даних моніторингу	–	–	–	–	–

Пропонується підхід, який полягає в імітаційному моделюванні розподілених обчислювальних систем, що базується на обліку даних моніторингу, що використовуються для динамічної корекції параметрів моделі (рисунок 1.5): завдання через систему управління навантаженням (1) надходять на обробку в обчислювальну систему (2), інформація про статус виконання завдань надходить у БД (3). Статистичні дані моніторингу використовуються як вхідний поток для імітаційної моделі. Також на базі статистичних даних про завдання можна згенерувати новий потік вхідних даних для моделі (4, 5). Дослідник отримує результати моделювання та аналізує їх (6), далі він може змінити параметри та перевіряти нові гіпотези (7,8).

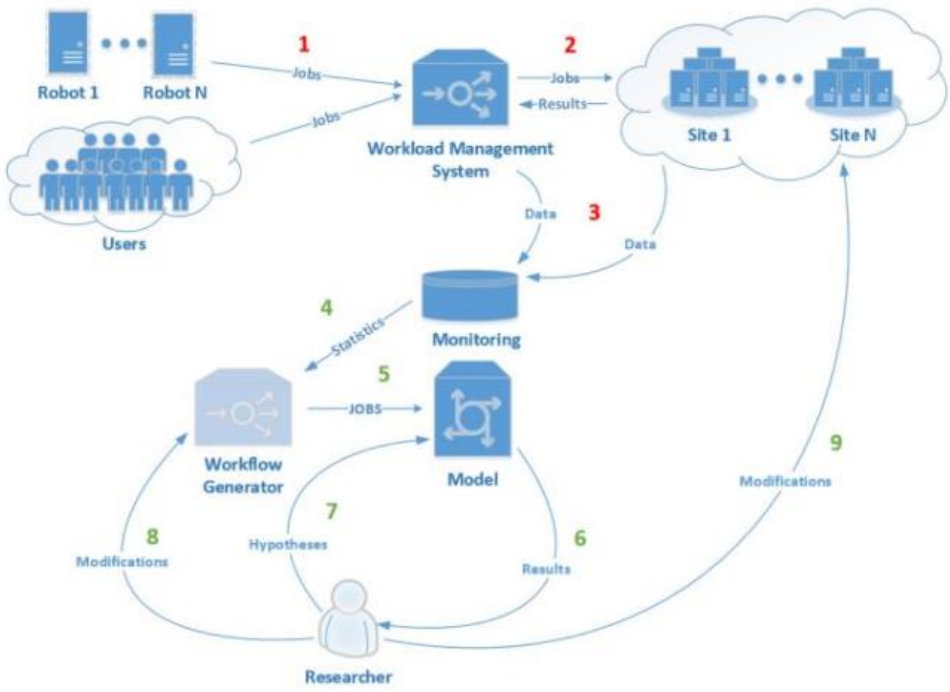

Рисунок 2.1 – Моделювання розподіленої системи з врахуванням даних моніторингу

Результати моделювання можуть бути використані для ініціалізації процедури зміни конфігурації ресурсного центру для поліпшення його

параметрів (9). Таким чином, модель повинна розглядатися як невід'ємна частина системи обробки даних, а дані моніторингу, як вхідні для моделювання. Ефективність цієї концепції у тому, що технічне рішення перевіряється на моделі перш, ніж обговорюється його фактична реалізація. Це дозволить приймати обґрунтовані проектні рішення при розвитку системи.

3 РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ ТА ОБЛІКУ РЕСУРСІВ У ГРІД–СИСТЕМІ

Система моніторингу та обліку ресурсів призначена для відстеження, поточного стану ресурсів, завдань та інших об'єктів у грід-системі. Серед основних завдань моніторингу зазначимо такі:

- безперервне спостереження за станом грід-сервісів;
- отримання інформації про обчислювальні ресурси (кількість обчислювальних вузлів для виконання завдань, архітектура обчислювальної системи, встановлене програмне забезпечення, доступні спеціалізовані програмні продукти), спожите процесорне час та інше;
- дані про доступ віртуальних організацій до ресурсів та використання ними квот на обчислювальні ресурси;
- моніторинг виконання обчислювальних завдань та завдань (запуск, зміна стану, коди завершення тощо).

Серед параметрів, необхідних для подальшого моделювання, найбільш суттєвими є такі дані моніторингу: задач (симуляція, аналіз, реконструкція) вступників до системи, обсяг використовуваної оперативної пам'яті, використаний процесорний час, число оброблених подій, час розрахунку задачі, обсяг даних, що використовуються.

3.1 Структура засобів моніторинга

Для реалізації системи моделювання потрібно було описати основні об'єкти та події грид та хмарних систем. Також потрібно розробити структуру бази даних, яка міститиме опис IT-інфраструктури, кожного її вузла, зв'язків між ними, дані моніторингу роботи різних підсистем та результати моделювання. Окрім 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 GSim 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 їх статистичний аналіз, результати якого дозволяють потім генерувати потік завдань зміни параметрів моделювання. Система, що моделюється, складається з блоків, склад і зв'язків, які показано на рисунку 3.1.

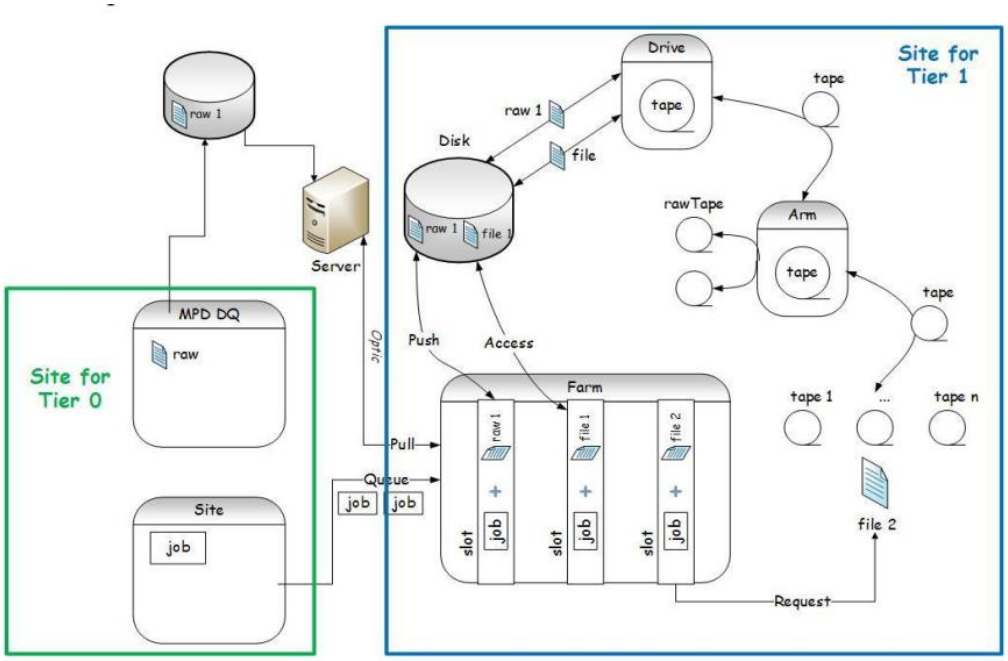

Рисунок 3.1 – Схема проходження завдань в GSim

Представлена структура в головних рисах відображає завдання, які вирішуються при накопиченні інформації з її паралельною обробкою, та відповідає структури центру рівня Tier1. Розроблений програмний інструментарій GSim дозволяє моделювати обробку потоку завдань IT-інфраструктурою, що має заданими ресурсами та правилами їх резервування та використання. Базовий функціонал імітаційної моделі реалізований шляхом доопрацювання та розширення класів системи моделювання

GridSim. 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 під час моделювання, які резерви має обчислювальна установка, тобто яка верхня межа інтенсивності потоку завдань (даних) та які компоненти установки надають на неї значний вплив.

Засіб опису обчислювальної інфраструктури реалізовано як база даних із веб-інтерфейсом (рисунок 3.2). Опис присвоюється ідентифікатор, який користувач має вказати у параметрах запуску моделі. Модель зчитує інформацію з бази та будує в пам'яті опис обчислювальної структури. Іншими даними є характеристики потоку завдань, що підлягають обробці, та інформація.

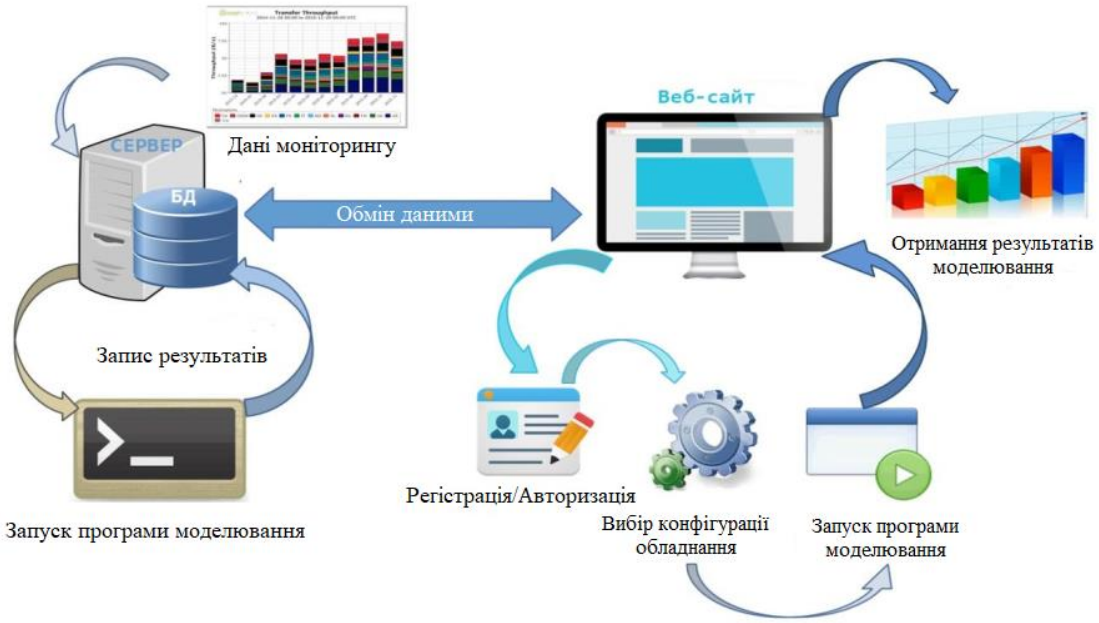

Рисунок 3.2 – Схема роботи GSim

Блок опису потоку завдань – набір утиліт, що дозволяє статистично проаналізувати результати моніторингу та сформувати потік завдань, аналогічний або відрізняється від проаналізованого на керований вплив користувача у вигляді впорядкованої за часом послідовності записів у базі даних.

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. Інший суттєвий за обсягом набір класів описує міграцію даних у системі. На відміну від існуючих рішень, де процес передачі даних реалізований на рівні пакетів, в GMSim передача даних реалізована через моделювання послідовності подій маніпуляції із файлами запит – відкриття – передача – закриття.

При цьому розраховується рівень навантаження на середовище передачі, а чи не час надходження пакета на вузол. Процес імітаційного моделювання полягає у проходженні набору завдань через IT-інфраструктуру.

Об'єктами моделі є завдання, процесори, файли, стрічки, дискові сховища, лінії передачі даних, 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. реалізованих класів дозволяє імітувати всі процеси, що відбуваються в системі. Для моделювання роботизованої бібліотеки було розроблено класи, представлені рисунку 3.3. 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.

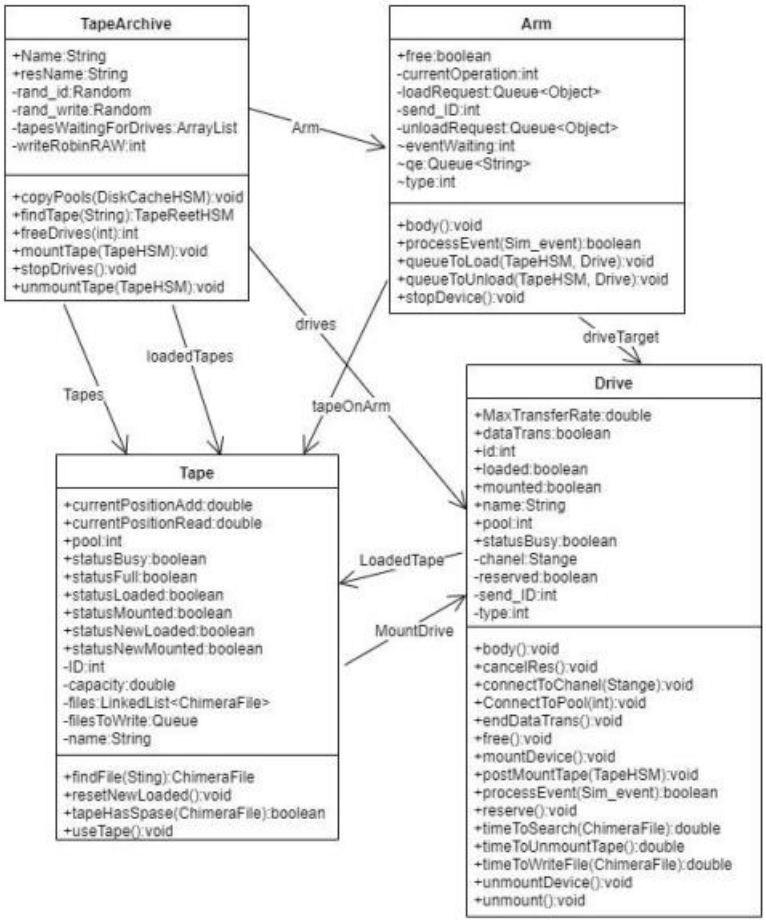

Рисунок 3.3 – Реалізація класів бібліотеки

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 єдиного файлу, копія якого може перебувати на дисковому масиві, чи стрічці. Завдання починає виконуватись, 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 в себе операцію приміщення стрічкового картриджа (tape) на драйв (drive), яку виконує рука робота (arm), монтування файлової системи картриджа на драйву та 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 центри 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 даних для моделювання. За відсутності необхідної статистики щодо обчислювальному центру чи моделюванню нового послідовність виконання обчислювального експерименту буде таке: 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 є послідовність записів у базі даних, що відображає всі події, що відбуваються у системі. До них відноситься, наприклад, надходження завдання, початок та кінець обробки завдання, початок та кінець передачі файлу,76 маніпуляції зі стрічками тощо. буд. Усі події описуються у єдиному форматі. Запис прив'язується до внутрішнього часу. Перед проектувальниками обчислювальних установок природним чином постає питання, як забезпечити масштабованість, тобто здатність установки зберігати працездатність при збільшенні потоків даних та задач.

У GMSim налаштування топології установки та характеристик її вузлів конфігурується через зовнішні параметри, що дозволяє легко налаштовувати модель під різні варіанти потоків. Найбільш простий параметр при налаштування – це кількість лічильників. Проте стверджувати, що швидкість обробки потоку завдань лінійно залежатиме від нього, неправильно. На швидкість впливатимуть також пропускна здатність каналів і продуктивність устаткування, у якому зберігаються вихідні дані. Можливість обліку цих факторів закладена у GMSim. При створенні моделі слід також звертати увагу на передбачуваний термін служби устаткування. Наприклад, рахункові

вузли можуть замінюватися в протягом кількох років, а роботизовані бібліотеки використовуються кілька п'ятиріч. Отже, на моделюванні процесів у них має бути зроблений особливий акцент.

3.2 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 наук експериментів. Існують різні системи управління завданнями та даними, наприклад, PanDA, DIRAC.

PanDA – система управління навантаженням розподіленому гетерогенному обчислювальному середовищі, що реалізує високопотокову концепцію обробки даних. Ця система останнє час розглядається як перспективна для кількох нових або модернізованих експериментів. Тому проектування системи моделювання та пов'язаної БД здійснювалося на підставі результатів аналізу проходження завдань у інфраструктури під керуванням системи PanDA.

Реальна грид система працює наступним чином:

- є сервер, якому з БД надсилаються завдання;
- до сервера звертається пілот, повідомляє сайт, на якому він запустився, і 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 завдання, статус завдання, витрачений комп'ютерний час, число подій, джерело завдання;

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

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 СУБД, що вільно розповсюджується. PostgreSQL.

Було розроблено структуру БД, зображену рисунку 3.3.

БД включає, наприклад, такі таблиці:

- Users – таблиця, яка містить інформацію про користувача: логін та пароль для входу в систему, електронна адреса;
- Experiments – таблиця, яка містить інформацію про наявні експерименти;
- Users_Experiments – таблиця, яка містить інформацію про зв'язок користувачів із експериментами;
- Simulation_Parameters – таблиця, яка містить описи 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 структури (диски, стрічки тощо);

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

-

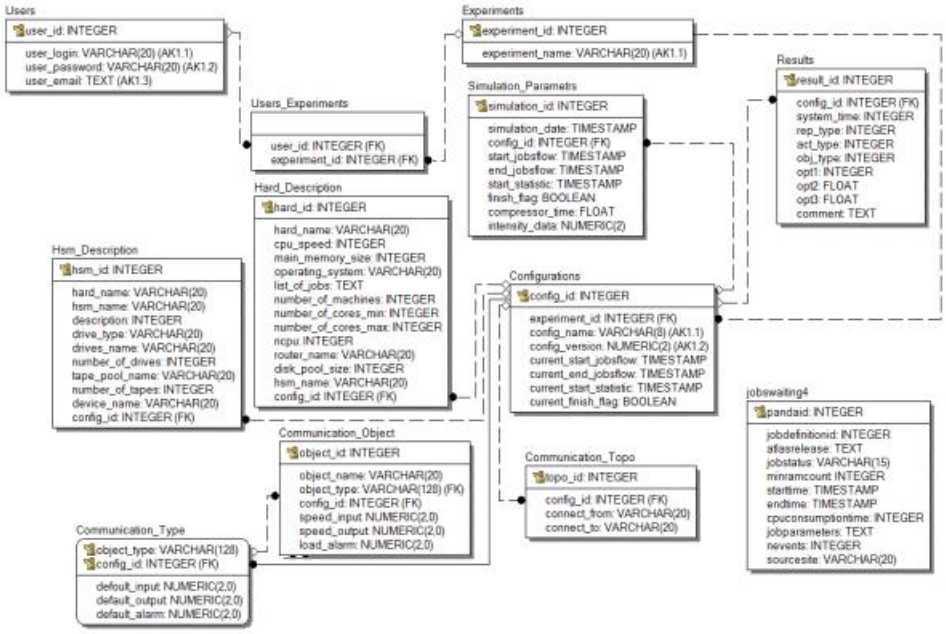

Рисунок 3.4 – Частина структури БД

Завантаження опису моделюваної обчислювальної структури реалізовано через веб-інтерфейс. Генерація потоку завдань здійснюється наступним чином (рисунок 3.5). Задаються необхідні параметри потоку завдань: період; перелік сайтів-джерел; кількість файлів у системі; максимальне та мінімальна кількість завдань на день для кожного із трьох типів (моделювання, реконструкція, аналіз); максимальне та мінімальне процесорний час, кількість подій та обсяг пам'яті).

Для кожного періоду випадково визначається кількість завдань.

Обчислюються випадкові величини: кількість процесорного часу, пам'яті та подій для завдання, час старту, а також генеруються Додаткові параметри.

Рисунок 3.5 – Інтерфейс GSim

Згенерований таким чином потік завдань зберігається у БД. Після чого користувач задає необхідну конфігурацію системи зберігання та запускає програму моделювання. Потім необхідно вибрати об'єкти, статистика про роботу яких буде записуватися в БД.

Модель запускається на сервері. Для цього було написано скрипт, який отримує необхідну інформацію з БД, а потім за допомогою командного рядку запускає модель.

Після завершення моделювання можна переглянути результати (рисунок 3.6). Для цього слід вибрати номер експерименту з випадаючого списку. На сторінці з'являться графіки, що описують різні характеристики системи, яка моделюється. Користувач може експортувати графіки та проводити подальший аналіз.

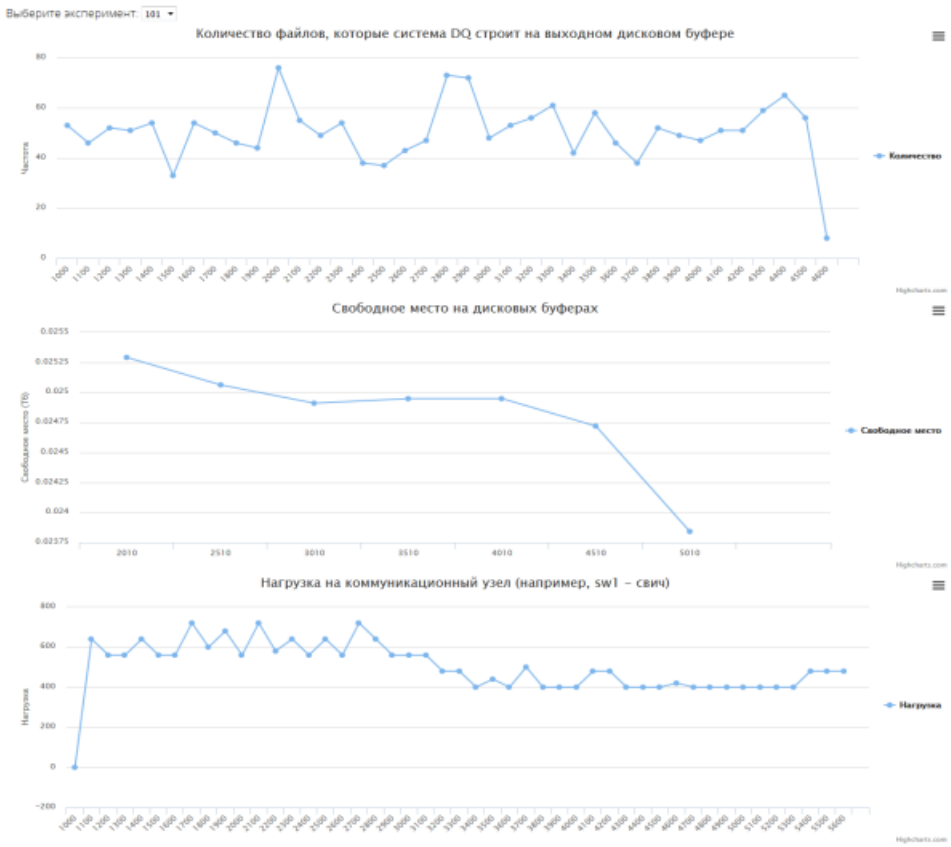

Рисунок 3.6 – Результати моделювання

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 здатність мережі, дані щодо навантаження 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 на неї істотно впливають. Таким Таким чином, GMSim дозволяє провести попередні дослідження варіантів IT-інфраструктури та оцінити можливості існуючої архітектури під час вирішення завдань зберігання та обробки даних.

ВИСНОВКИ

У кваліфікаційній роботі проаналізовані методи та засоби моделювання систем обробки даних. Був розроблений та реалізований програмний інструментарій GSim для моделювання систем зберігання та обробки даних, що реалізує синтез процесів моделювання та моніторингу, який дозволяє провести попередні дослідження різних варіантів організації IT-інфраструктури, оцінити можливості існуючої архітектури при вирішенні завдань 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 здатність мережі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Новіков В.С., Дяченко В.О., Носик А.М. Аналіз методів моделювання розподілених систем обробки даних // Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління. Тези доповідей дванадцятої міжнародної науково-технічної конференції 27-28 квітня 2022 р., т.1. Баку-Харків-Жиліна. 2022 р. С.75.

2. Кархан Эрджиес. Распределенные системы реального времени. Теория и практика / ДМК Пресс, 2020. С. 382.

3. 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.

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

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.

7. Cartlidge J., Cliff D. Comparison of cloud middleware protocols and subscription network topologies using CReST, the cloud research simulation toolkit // Proceedings 3rd Int. Conf. Cloud Computing and Services Science. – Germany: SciTePress. – 2013. – P. 58-68.