ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Пояснювальна записка

до кваліфікаційної роботи на здобуття ступеня вищої освіти магістр

на тему: **«Модель планування завдань у Grid системі з кластеризованими ресурсами»**

Виконав: здобувач вищої освіти
за освітньо-професійною програмою
Інформаційні управляючі системи та технології
спеціальності 126 Інформаційні системи та технології
ступеня вищої освіти магістр
групи 126ІСТ_мд_22
Курянчик А.С.
Керівник: Поночовний Ю.Л.
Рецензент: Брикун О.М.

Полтава – 2023 року

ВСТУП

Актуальність теми. Нині важко уявити отримання наукових результатів без використання обчислювальної техніки. В одних випадках для цього досить звичайних комп'ютерів, в інших же необхідно зробити велику кількість складних розрахунків, що вимагає великої кількості обчислювальних ресурсів. Для цього створюються обчислювальні комплекси, які географічно розподілені, але функціонують у інтересах однієї організації.

У зв'язку з цим набула популярності концепція розподіленої обчислювальної інфраструктури під назвою Grid [1]. У цій роботі розглядається Grid, що складається з невідчужуваних некластеризованих ресурсів (окремих комп'ютерів, які використовуються їх власниками) і орієнтований на обробку послідовних завдань, яким для виконання потрібен один процесор, а також серіалізованих завдань, тобто набору послідовних завдань, які вирішують одне завдання, але не взаємодіють між собою у процесі виконання.

Слід зазначити, що найбільшого поширення набув метод побудови Grid інфраструктур із кластерних вузлів, у яких об'єднується багато комп'ютерів, що належать одному адміністративному домену – дворівневий спосіб організації [2]. За такої організації кластери ресурсів знаходяться під управлінням локального менеджера, який здійснює безпосередній розподіл завдань та їх запуск на обчислювальних установках.

Головна відмінність дворівневого та однорівневого грідів полягає в режимі використання ресурсів, від якого суттєво залежать способи управління інфраструктурою загалом та окремими виконавчими комп'ютерами [3]. Поняття ресурсу в гріді є дуже широким і обмежується лише системними ресурсами комп'ютерів (процесор, пам'ять, дисковий простір). Ресурсом також може бути, наприклад, пристрій, підключений до мережі, а також будь-яка програма, яка з будь-яких причин не може бути встановлена у всіх, хто хоче обробляти з його допомогою свої дані. Тому питання побудови моделей планування завдань у Grid системі для ефективного використання кластеризованих ресурсів є актуальним.

Зв'язок роботи з науковими програмами, темами. Робота відповідає дослідженням в межах науково-дослідної роботи «Розвиток підприємництва: управлінські, економічні, інноваційна та правові аспекти» відповідно до договору №9 від 15.05.2023 р. між ТОВ «ПАФ Гарант» та Полтавським державним аграрним університетом (розділ «Обгрунтування показників оцінювання гарантоздатності розподілених інформаційних систем»).

Метою кваліфікаційної роботи є підвищення ефективності використанні ресурсів у Grid системі з кластеризованими ресурсами за рахунок управління плануванням завдань.

Завданнями кваліфікаційної роботи є:

– аналіз рішень для управління розподіленими обчислювальними ресурсами;

– проєктування архітектури системи диспетчеризації завдань у гріді;

– розробка моделі функціонування Grid системи з кластеризованими ресурсами для планування завдань.

Об'єктом дослідження є процеси функціонування Grid систем з кластеризованими ресурсами.

Предметом дослідження є модель Grid системи з кластеризованими ресурсами для організації планування завдань.

Методи дослідження – проведені в роботі дослідження базуються на методах теорії ймовірності, системного і марковського аналізу, систем масового обслуговування, які використовувалися при розробці марковських моделей функціонування Grid системи з кластеризованими ресурсами.

Інформаційна база кваліфікаційної роботи складається з наукових статей, міжнародних аналітичних видань і звітів, матеріалів наукових конференцій інтернет-ресурсів, що місять інформацію про архітектуру сучасних Grid систем, а також даних, отриманих від провідних IT-компаній у сфері розподілених обчислень.

Елементи наукової новизни полягають у розроблені та досліджені аналітичної моделі функціонування Grid системи з кластеризованими ресурсами в умовах відмов в обслуговуванні через зайнятість ресурсів.

Практична значущість роботи полягає в можливості повторного застосування та модифікації розробленого програмного коду моделі для оцінювання показників якості обслуговування розподілених обчислювальних систем з відмовами і обмеженими чергами обслуговування, а також показників їх економічної ефективності. Отримані результати можуть бути корисними для ІТ фахівців при моделюванні спеціалізованих інформаційних управляючих систем.

Апробація результатів дослідження відбувалася шляхом оприлюднення доповідей на наукових конференціях, семінарах.

Публікації. За результатами проведеного дослідження опубліковано тези: «Організація та планування завдань у віртуалізованих системах для розподілених обчислень», Матеріали XII Міжнар. наук. конференції «Інформаційні технології в енергетиці та агропромисловому комплексі», м. Львів, 04-06 жовтня 2023 р.

Структура та обсяг кваліфікаційної роботи логічно пов’язані з задачами досліджень. Робота містить перелік умовних позначень, вступ, три розділи основної частини, висновки, список використаних джерел, додатки. Загальний обсяг текстової частини дипломної роботи складає 61 сторінка формату А4. Вона містить 13 рисунків і 3 таблиці. У роботі використано 43 науково-технічних джерела.

РОЗДІЛ 1
АНАЛІЗ РІШЕНЬ ДЛЯ УПРАВЛІННЯ РОЗПОДІЛЕНИМИ ОБЧИСЛЮВАЛЬНИМИ РЕСУРСАМИ

1.1 Використання однорівневої організації розподілених комп'ютерів на прикладі проєктів BOINC

Сьогодні практика об'єднання персональних комп'ютерів досить поширена. Завдання об'єднання вирішується як і географічно розподіленому середовищі, і у межах одного підприємства. При організації подібних інфраструктур можна виділити чотири підходи: перший ґрунтується на створенні проєктів, до яких підключаються комп'ютери – виконавці; другий підхід полягає у застосуванні P2P технологій (Peer-to-Peer) [4] та об'єднанні виконавчих комп'ютерів в однорангові мережі; третій підхід характеризується системами із централізованим управлінням; четвертий підхід представлений приватними розробками корпоративних систем.

Слід зазначити, що цей поділ є умовним і покликаний показати різноманіття технологій, що використовуються, а також широту застосування подібних систем. Так, аналізовані далі системи можуть перетинатися з різних аспектів реалізації чи за архітектурним рішенням.

Один з перших випадків використання однорівневої організації розподілених комп'ютерів був проєкт SETI@Home від Каліфорнійського університету, який спрямовувався на пошук позаземного інтелекту [5]. Учасникам цього проєкту пропонувалося завантажити невелику програму з вебсайту університету, яка працювала, коли комп'ютер не використовувався власником, замінюючи екран зберігання. Ця програма отримувала дані з радіотелескопічних спостережень, аналізувала їх і відправляла результати назад. Радіотелескопи генерували величезний потік інформації, який оброблявся невеликими частинами за допомогою програми. Програма рідко змінювалась, тому вона завантажувалась на виконавчий комп'ютер один раз та оновлювалась за потреби.

Пізніше з'явилося багато проєктів серії @Home, які призначені для

вирішення подібних завдань, проте SETI@Home залишається одним з найпопулярніших. У липні 2007 року в ньому брали участь 658 тисяч користувачів [6], а загальна обчислювальна потужність комп'ютерів становила 262 Tflops. Ці зусилля університету призвели до створення програмної платформи BOINC [7], яка спрощує створення нових проєктів (рис. 1.1).

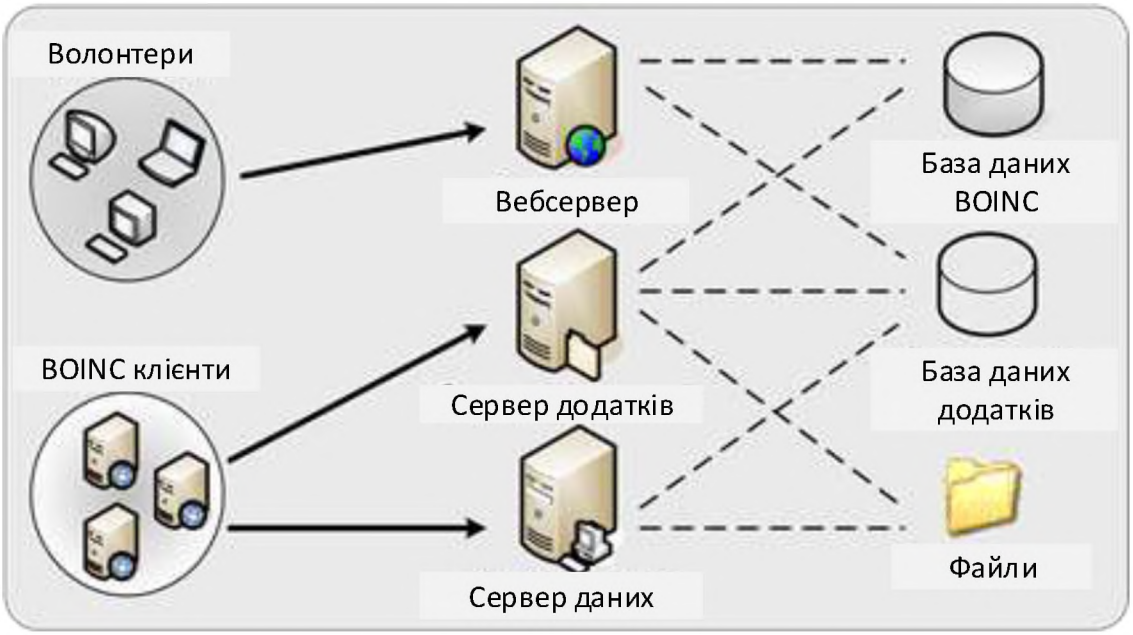

Рисунок 1.1 – Обчислювальна інфраструктура під керуванням системи BOINC

Розподілені обчислення у системі BOINC ґрунтуються на проєктах. Кожен сервер BOINC може підтримувати декілька незалежних проєктів, кожен з яких призначений для виконання конкретної програми та має свою програмну та апаратну інфраструктуру. Процес створення сервера та запуску проєкту включає кілька послідовних кроків:

1. Встановлення програмного забезпечення: BOINC базується на загальнопоширеному базовому програмному забезпеченні, яке потрібно встановити та налаштувати перед роботою. Для зберігання різноманітної інформації (користувачі, виконавчі комп'ютери, підтримувані платформи, блоки вхідних даних, додатки тощо) використовується СУБД MySQL [8]. Доступ користувачів до інформації проєкту здійснюється через вебінтерфейс. Для налаштування BOINC-сервера також потрібно встановити Python (для запуску

конфігураційних скриптів), PHP (для генерації html сторінок вебінтерфейсу) та libcurl (для передачі даних).

2. Встановлення ПЗ BOINC-сервера та створення проєкту: Сервер встановлюється шляхом збирання з вихідних кодів. Потім запускається скрипт для створення проєкту та визначається його ім'я. Для нового проєкту створюється структура директорій і база даних, де зберігається інформація про проєкт.

3. Встановлення та налаштування вебсервера: BOINC використовує вебсервер Apache, який забезпечує доступ користувачів до інформації проєкту через вебінтерфейс, взаємодію сервера з виконавчими комп'ютерами та передачу вхідних порцій на виконавчі комп'ютери та результатів обчислень на сервер.

4. Конфігурування проєкту: Формується конфігураційний файл з типовими налаштуваннями для більшості проєктів. Потрібно відредагувати значення деяких параметрів, таких як ім'я користувача та пароль для доступу до бази даних проєкту. Налаштовують демон, який генерує вхідні дані, та планувальник, який розподіляє ці дані по виконавчих комп'ютерах.

5. Створення та підготовка програми для запуску: Важливим моментом є створення програми, яка відповідає вимогам системи BOINC. Ця програма запускається на всіх виконавчих комп'ютерах і взаємодіє з API BOINC. Написана програма може бути скомпільована для різних платформ, щоб автоматично розподіляти її на виконавчі комп'ютери з відповідною архітектурою.

При створенні власного BOINC-сервера для управління інфраструктурою проєкту, організація публікує інформацію про нього з метою привернення добровольців для вирішення конкретного завдання. Ця інформація містить опис проєкту та мережеву адресу, звідки можна завантажити клієнтську програму, що обов'язково зазначається в налаштуваннях клієнтської програми.

Щоб долучитися до проєкту, власник комп'ютера реєструється на сайті проєкту та отримує персональний ключ, який використовуватиметься для ідентифікації при подальших підключеннях. Після підключення сервер визначає архітектуру виконавчого комп'ютера та надає відповідний екземпляр програми разом з порцією вхідних даних.

Подальша доставка програми та вхідних даних на комп'ютер-клієнт здійснюється, і програма починає свою роботу тоді, коли власник знижує активність. Після завершення обробки отриманої порції даних, клієнт надсилає результати на сервер і отримує чергову порцію. Коли всі порції даних оброблені, отримані від клієнтів часткові результати об'єднуються на сервері, і виконання проєкту припиняється.

1.2 Аналіз переваг та недоліків систем Cluster Computing On the Fly та OurGrid

Прикладом системи, що використовує обчислювальні ресурси окремих комп'ютерів, є розробка Cluster Computing On the Fly (CCOF) від Орегонського університету [9, 10]. Архітектура цієї системи ґрунтується на підході P2P та схожа на файлообмінні мережі, з тією відмінністю, що виконавчий комп'ютер виступає ресурсом замість файлів. Основні компоненти цієї програмної архітектури включають в себе планувальника користувача, який працює на їхньому комп'ютері, та планувальника виконавчого комп'ютера. Система управляє розподіленою інфраструктурою виконавчих комп'ютерів, координуючи планування через їхніх планувальників.

Як і в будь-якій P2P-мережі, усі комп'ютери тут є рівноправними, відсутній центральний сервер та клієнти [11]. Кожен комп'ютер виступає зв'язковим елементом між іншими (рис. 1.2) та зберігає інформацію про ресурси сусідів. Навіть якщо власник комп'ютера активно використовує його ресурси, і надходження зовнішніх завдань припинено, комп'ютер продовжує приймати та передавати повідомлення через себе. Він також надає решті учасників шаблон, що вказує на зайнятість ресурсів протягом певних інтервалів часу.

Визначення вільних ресурсів є основною складовою представлених шаблонів. Існує декілька алгоритмів пошуку виконавчих комп'ютерів, які базуються на тому, що після отримання ресурсного запиту від користувача,

планувальник комп'ютера перевіряє його за шаблоном, щоб визначити можливість виконання завдання. Якщо запит відповідає шаблону і комп'ютер вільний, користувач передає завдання йому.

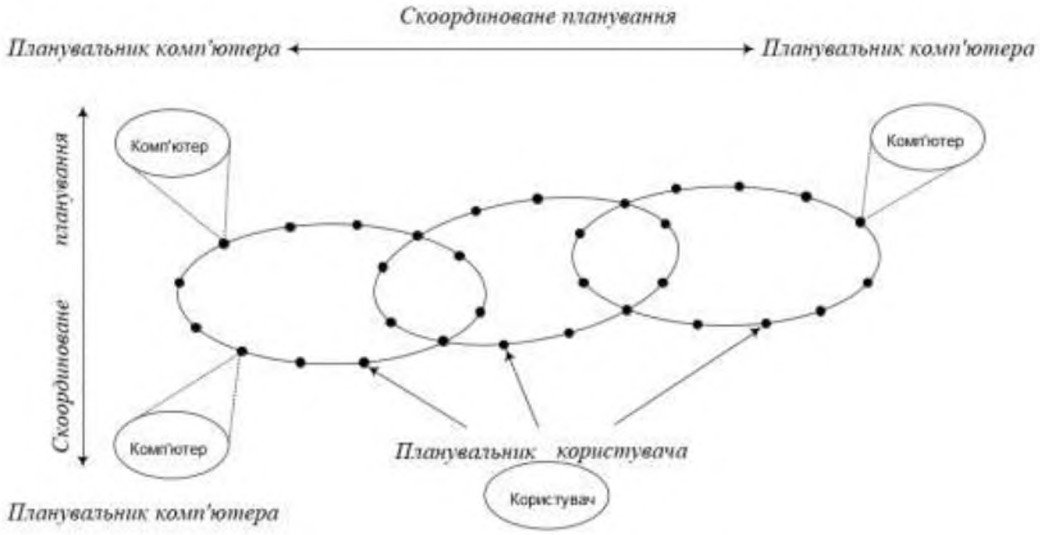

Рисунок 1.2 – Архітектура системи CCOF [12]

Під час виконання завдання планувальник користувача отримує періодичні сигнали від виконавчого комп'ютера про його активність, продовжуючи виконання завдання. У випадку виникнення помилки на виконавчому комп'ютері, сигнали припиняються, а планувальник користувача перенаправляє своє завдання на інший виконавчий комп'ютер.

Власне, розподіл завдань здійснюється планувальником користувача на його комп'ютері. Його запити оцінюються планувальником виконавчого комп'ютера, який вибирає певні завдання залежно від локальних політик (таких як продуктивність, рейтинг, рівень довіри тощо). Таким чином, система CCOF реалізує децентралізовану схему планування, що має як позитивні, так і негативні аспекти, притаманні цьому підходу.

Система OurGrid [13, 14] спробувала розробити універсальний підхід, придатний для створення розподілених інфраструктур, побудованих як з кластерів, так і окремих комп'ютерів. Крім того, OurGrid поєднує централізоване розподілення ресурсів з децентралізованою архітектурою, заснованою на P2P-мережах.

Ресурсна інфраструктура поділяється на частини, кожна з яких є сайтом і складається з комп'ютерів, що належать до одного адміністративного домену. Ці комп'ютери можуть бути як персональними, так і частиною кластера, який керується менеджерами ресурсів, такими як PBS і LSF.

У системі реалізовано три типи компонентів: диспетчер сайту, компонент виконавчої машини та інтерфейс користувача (рис. 1.3). Для керування сайтом існує окремий комп'ютер, на якому встановлюється програмний компонент OurGrid peer – диспетчер сайту [15]. Компонент SWAN (Sandboxing Without A Name) розміщується на всіх виконавчих комп'ютерах кожного сайту і відповідає за безпосередню обробку завдань. Користувачі взаємодіють з системою через персонального брокера MyGrid, який вони встановлюють на свої комп'ютери.

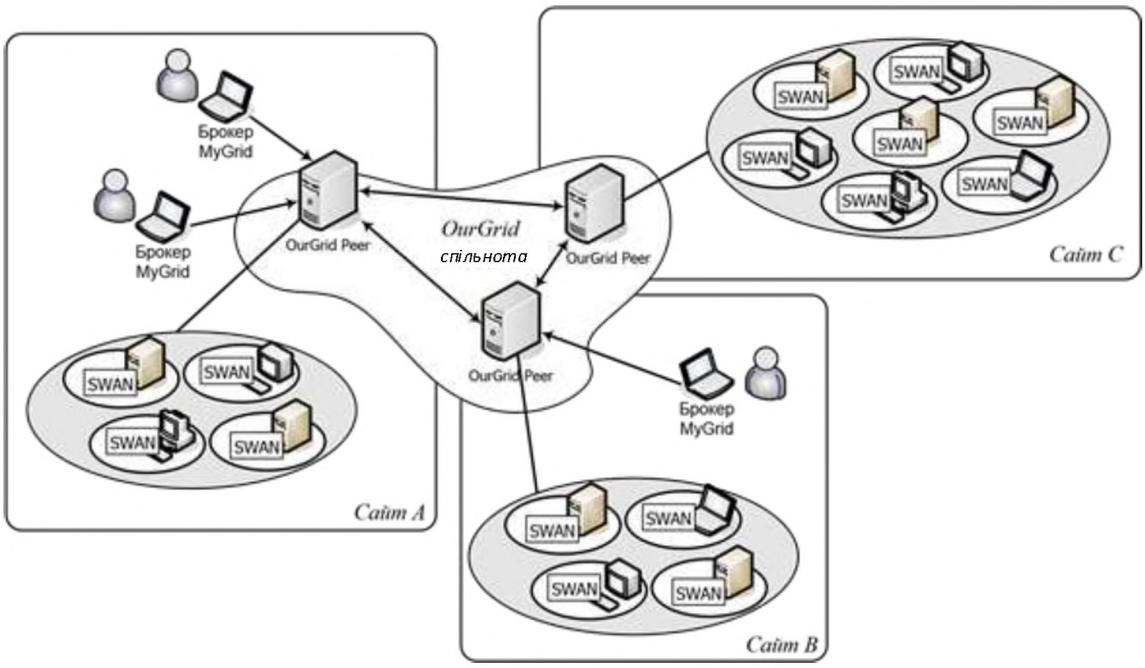

Рисунок 1.3 – Архітектура системи OurGrid [14]

Брокер MyGrid здійснює розподіл завдань за ресурсами та надає абстракції, що приховують гетерогенність обчислювального середовища від користувача. Завдання користувача передаються брокеру, який запитує необхідні ресурси на всіх сайтах, де користувач має обліковий запис (account). Кожен сайт керується своїм диспетчером (OurGrid peer), який повідомляє брокеру, які ресурси доступні користувачеві. Ці ресурси можуть бути локальними (частиною сайту, що

контролюється диспетчером) або зовнішніми, які належать іншим сайтам. Комп'ютери, де встановлено компонент OurGrid peer, утворюють P2P-мережу. Диспетчер отримує інформацію про зовнішні ресурси, взаємодіючи з диспетчерами інших сайтів у P2P-мережі аналогічно до системи CCOF.

Спосіб планування, використаний у брокері, орієнтований на послідовні завдання. Брокер розподіляє частини послідовних завдань, використовуючи критерій мінімізації часу виконання завдань в цілому. Щоб гарантувати завершення завдання, брокер використовує механізм міграції, розглядаючи можливість відключення виконавчих комп'ютерів.

Служба безпеки SWAN використовує монітор віртуальних машин Xen [16], який ізолює код користувача програми всередині оболонки (sandbox). Завдання, що виконуються всередині цієї оболонки, не мають доступу до реальних комп'ютерних даних і не можуть використовувати мережу. Система підтримує обробку завдань, що не потребують обміну даними під час виконання. Хоча спочатку систему OurGrid розробляли для грідів, що складаються з кластерів, зараз її можна використовувати для організації обчислень в однорівневому Grid. Для цього кожен виконавчий комп'ютер повинен мати, крім встановленого компонента SWAN, також диспетчер OurGrid peer.

1.3 Механізми Grid систем з централізованим управлінням

Система метакомп'ютингу X-Com [17, 18] розроблена для організації розподілених обчислювальних середовищ, де можна виконувати різноманітні завдання. Розробники системи надали інструментарій, що дозволяє користувачеві запускати програми на різних обчислювальних вузлах. Система забезпечує контроль за виконанням програм, балансуванням навантаження між обчислювальними вузлами та візуалізацією процесу виконання.

Для підготовки програми для паралельного виконання у системі X-Com, користувачеві необхідно провести деякі модифікації, зокрема, виділити серверну

та клієнтську частини програми. Основні особливості системи X-Com включають можливість роботи в глобальній мережі, що дає можливість підключати велику кількість комп'ютерів, а також використання різноманітних ресурсів – від кластерів до звичайних робочих станцій з різними програмно-апаратними платформами.

Система X-Com складається з трьох рівнів архітектури: користувача, серверного та виконавчих комп'ютерів. На рівні користувача розташовані процеси, що взаємодіють з системою, дозволяючи користувачеві поставити завдання у чергу та відстежувати їх стан.

Рівень виконавчих комп'ютерів представлений компонентом виконавця X-Com, який отримує від сервера відповідного завдання порції вхідних даних, запускає обчислювальну частину програми користувача та повертає результати на сервер. Процеси на серверному рівні включають «сервер завдання», «керуючий сервер» та «менеджер виконавців». Ця група процесів відповідає за підтримку виконання завдань користувача [19]:

– сервер завдання забезпечує виконання конкретної програми користувача – він взаємодіє з серверною частиною програми, яка генерує порції вхідних даних та обробляє результати, керує логікою видачі порцій виконавцям та взаємодіє з ними через допоміжні служби (файлову, запиту та результату) та статистики;

– керуючий сервер відповідає за керування чергами завдань, породжує свої сервери завдань для кожного завдання та надсилає команди відповідним клієнтам;

– менеджер виконавців приймає запити виконавців та перенаправляє їх до серверів завдань, що відповідають характеристикам комп'ютера.

Система X-Com використовує Інтернет для взаємодії між комп'ютерами та сервером, використовуючи лише стандартний протокол HTTP, що дозволяє підключати практично будь-які обчислювальні потужності, що мають доступ до Інтернету. Дані, що передаються системою, аналогічні звичайному Інтернет-трафіку, і не вимагають налаштування проксі-сервера, firewall або інших систем захисту.

Для оптимальної взаємодії та підключення більшої кількості комп'ютерів

обчислювальна інфраструктура, керована системою X-Com, може бути структурована як ієрархічне дерево. Виконавчі комп'ютери завжди будуть листям цього дерева, тоді як центральний сервер X-Com залишиться в корені структури. Проміжні сервери виглядають як звичайні комп'ютери для центрального сервера, а для виконавчих комп'ютерів вони представляються як центральні сервери.

Система X-Com подібна до платформи BOINC, але вона має відмінності, які усувають певні обмеження платформи BOINC. Наприклад, у системі X-Com може одночасно працювати безліч додатків на одній програмно-апаратній інфраструктурі, у той час як для кожної нової програми у BOINC необхідно створювати власну інфраструктуру. Керуючий сервер у системі X-Com створює окремий сервер завдання для нових завдань із відповідним файлом налаштувань.

Для кожної прикладної програми необхідно ретельно адаптувати та представити її у вигляді двох частин – серверної та клієнтської. Серверна частина відповідає за підготовку вхідних даних, які обробляються клієнтською частиною на обчислювальних вузлах. Можна використовувати попередньо підготовлені дані, які подаються у вигляді окремих файлів. У такому разі необхідно належним чином описати ці дані у конфігураційному файлі та забезпечити їхню доставку на сервер.

Запуск завдань управляється підсистемою управління завданнями, яка отримує завдання користувачів, формує чергу та послідовно передає їх виконання на виконавчі комп'ютери. Також ця підсистема забезпечує відстеження стану запущених завдань. Якщо завдання може виконуватися на комп'ютерах з різними програмно-апаратними платформами, для кожної з них на сервері повинен існувати відповідний файл клієнтської частини програми.

У системі X-Com розподіл завдань по комп'ютерах відбувається за методом, за яким на комп'ютер, який запитує завдання, приходить випадкове відповідне за апаратними вимогами завдання. При цьому не враховуються час виконання завдання на комп'ютері та його обмежений термін виконання.

Під час підготовки завдання користувач формує файл опису завдання, в якому вказуються такі дані [20, 21]:

– ім'я завдання;

– ідентифікатор;

– параметри командного рядка для запуску клієнтської частини програми;

– вимоги до ресурсів, на яких може виконуватися завдання;

– адреси серверів, з яких слід завантажувати файли завдань.

Клієнтська частина X-Com збирає та надсилає серверу дані про ресурси обчислювального вузла, які включають наступну інформацію:

– тип операційної системи;

– апаратну архітектуру процесора;

– продуктивність процесора;

– об'єм оперативної пам'яті.

Для визначення можливості вузла виконувати завдання у системі X-Com порівнюють опис завдання з описом вузла. Дані, які потрібні для виконання завдання у системі X-Com, поділяються на два типи: файли завдання (виконані файли та файли даних) і порції вхідних даних для рахунку. Файли першого типу можуть бути розташовані на доступному http сервері, звідки клієнтська частина X-Com завантажить їх на обчислювальний вузол перед початком виконання завдання. Файли з порціями даних для рахунку мають бути на сервері до запуску завдання. Їх можна доставити вручну заздалегідь та спеціально описати, або згенерувати за правилами серверної частини програми.

Для подальшої обробки завдання у системі X-Com використовується загальноприйнятий механізм роботи з додатками, що серіалізуються. Після завантаження всіх необхідних файлів на обчислювальний вузол, клієнт звертається за порцією даних для рахунку та запускає процес обробки. Коли порція даних оброблена, клієнт надсилає результат на сервер і отримує чергову порцію. Цей процес триває до тих пір, поки не будуть оброблені всі порції.

У системі X-Com є засоби захисту від різних загроз, таких як [22, 23]:

– модифікація мережевого трафіку, що порушує конфіденційність та достовірність даних;

– підміна однієї зі сторін передачі даних, що дозволяє зловмиснику атакувати

вузли інфраструктури;

– використання помилкового клієнта для отримання доступу до даних, призначених для обробки, та відправлення недостовірних результатів.

Для розв'язання цих проблем використовуються механізми шифрування трафіку та ідентифікація серверів та клієнтів за їх сертифікатами згідно зі стандартом X.509 [24].

1.4 Загальні вимоги до програмного забезпечення Grid систем

Серед ключових вимог до програмного забезпечення гріду, яке поєднує кластеризовані ресурси, можна виділити наступні аспекти:

1. Запуск та керування завданнями на кластеризованих ресурсах: ПЗ Grid має забезпечувати виконання завдань на широкому спектрі ресурсів. Це передбачає приймання завдань у загальноприйнятій формі (наприклад, RSL [25] або JSDL [26]). Мови опису дозволяють враховувати специфіку гріду, таку як обмеження на час обробки завдань.

2. Підтримка управління завданнями: Програмні інтерфейси для запуску та керування повинні відповідати архітектурі гріду, що дозволяє доступ до Grid з будь-яких додатків, що потребують додаткових ресурсів.

3. Колективне використання та розподіл ресурсів: У процесі виконання завдання система має виділяти та контролювати використання ресурсів, доставку вхідних файлів та результатів.

4. Стандартизованість: ПЗ Grid повинно відповідати загальноприйнятим стандартам (наприклад, OGSA [27], WSDL [28], SOAP [29]), щоб забезпечити сумісність з різними Grid інфраструктурами.

5. Підтримка віртуальних організацій: ПЗ Grid повинно розмежовувати доступ до ресурсів між різними віртуальними організаціями.

6. Інтероперабельність: ПЗ Grid має надавати інформацію про ресурси інфраструктури у стандартному форматі для управління ієрархічно вкладеними

сегментами Grid.

Для Grid системи з невідчужуваними ресурсами, такі ресурси використовуються для обробки завдань Grid та локальних процесів. У цьому випадку система повинна дотримуватись пріоритетів, забезпечувати безпеку даних та ресурсів, а також враховувати динамічний характер Grid.

Висновки до розділу 1

У першому розділі проводиться огляд та аналіз наявних рішень для організації обчислювальних інфраструктур із кластеризованих комп'ютерів. Під час аналізу розглянуто чотири основних підходи: перший полягає у створенні проєктів, до яких підключаються комп'ютери – виконавці; другий використовує P2P-технології та об'єднання виконавчих комп'ютерів у однорангові мережі; третій характеризується системами із централізованим управлінням; а четвертий представлений приватними розробками корпоративних систем. Також у розділі сформульовані вимоги до програмного забезпечення для однорівневих Grid систем.

На основі проведеного аналізу сформульовані загальне завдання, яке полягає у розробці моделі функціонування Grid системи з кластеризованими ресурсами для оптимізації планування завдань. Загальне завдання розділено на три часткові задачі, дві з яких розглянуті у наступних розділах.

РОЗДІЛ 2
ПРОЄКТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ ДИСПЕТЧЕРИЗАЦІЇ ЗАВДАНЬ У GRID

2.1 Обґрунтування складу компонентів системи диспетчеризації

Для інтеграції різноманітних ресурсів і їх включення до складу Grid інфраструктури запропоновано архітектуру, що складається з системи диспетчеризації, яка включає три основні компоненти: диспетчера, агента та інтерфейсу користувача (рис. 2.1).

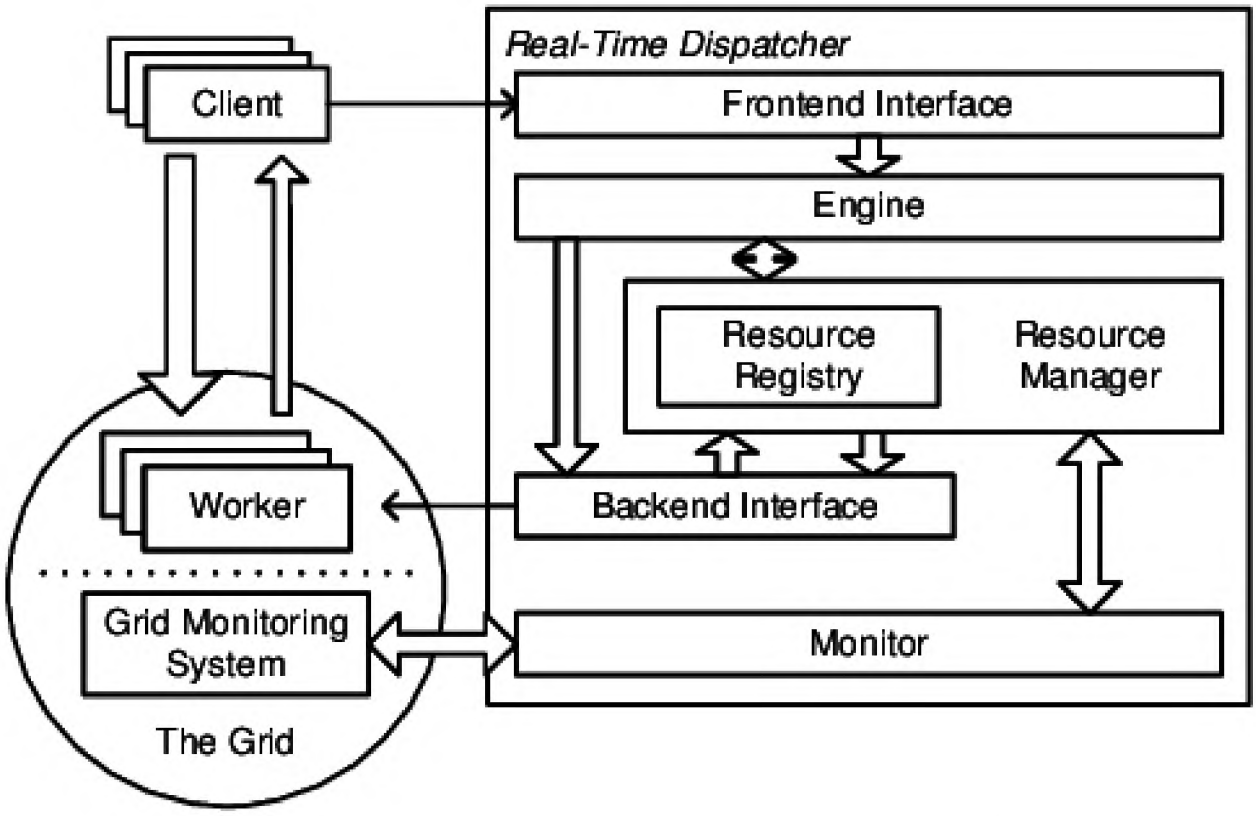

Рисунок 2.1 – Архітектура системи диспетчеризації [30]

Згідно з пропозиціями у дослідженні [6], виконавчі комп'ютери ресурсної інфраструктури не мають безпосереднього доступу для користувачів, тому зовнішній інтерфейс до них централізований у диспетчері. Диспетчер встановлюється на виділеному сервері у Grid сегменті, і до нього приєднуються

різні виконавчі комп'ютери, включаючи просторово розташовані. Для забезпечення сумісності з існуючими Grid системами диспетчер використовує стандартний інтерфейс запуску завдань (аналогічний до GRAM [31]), а реалізація інформаційної служби дозволяє публікацію даних про доступні ресурси сегмента у інформаційній системі інфраструктури.

Основне завдання агента, який встановлюється на виконавчому комп'ютері, полягає у керуванні завданнями під час їх виконання. Інтерфейс користувача дозволяє користувачам керувати завданнями у стандартний для Grid спосіб та отримувати інформацію щодо статусу завдання.

2.2 Завдання диспетчера однорівневого гріду

Диспетчер однорівневого Grid забезпечує доступ до сегменту Grid з розподілом завдань між комп'ютерами, забезпечуючи доступ до ресурсів. Диспетчер складається з низки Grid служб, які реалізують базові компоненти програмного забезпечення для Grid та забезпечують взаємодію з агентами.

Grid служби ґрунтуються на стандартах вебслужб, що робить їх універсальними для будівництва розподілених систем та розробки Grid додатків. Ця система включає такі служби [32, 33]:

- служба взаємодії з агентами;
- служба керування завданнями;
- планувальник;
- інформаційна служба;
- служба передачі.

Реляційна база даних використовується для зберігання інформації про користувачів, завдання та виконавчі комп'ютери, а також службову інформацію. У

системі реалізовано різні механізми обмеження споживання ресурсів комп'ютера. Наприклад, деякі перевірки можуть проводитися самим диспетчером на основі ресурсного запиту завдання, або ж агент може самостійно контролювати використання ресурсів виконавчого комп'ютера.

Служба взаємодії з агентами підтримує зв'язок між диспетчером та виконавчими комп'ютерами, вона відповідає за автоматичну реєстрацію нових комп'ютерів, обробку інформації про стан завдань та ресурси. Для точного планування доступності виконавчих комп'ютерів у гріді важливо враховувати три типи доступності: взаємодія з диспетчером у час, можливість запуску завдання на комп'ютері в даний час та обсяг обчислювальних ресурсів, доступних під час виконання завдання.

Окрім поточної доступності, для ефективного планування необхідно мати прогноз стану та обсягу ресурсів у майбутньому, що вимагає накопичення відповідної інформації під час періодичних звітів агентів.

Служба управління завданнями виконує прийом та обробку завдань, додаючи інформацію про них до інформаційної бази, а також виконує інші керуючі команди за запитами користувачів. Після прийому завдання користувач отримує ідентифікатор, що дозволяє керувати ним, відправляючи відповідні команди службі та отримуючи інформацію про стан та ресурси, використані завданням. Механізми авторизації забезпечують керування завданням тільки його власнику – зареєстрованому користувачеві, який вводить завдання в систему.

Служба планування (планувальник) відповідає за розподіл завдань між виконавчими комп'ютерами [34]. Система використовує пріоритетне планування черги завдань, спрямоване на максимізацію використання наявних ресурсів (комп'ютерів) та забезпечення завершення завдань у відведений час.

Вся необхідна інформація для планування зберігається у інформаційній базі. Первинна інформація про ресурси та завдання надходить до бази від служб

управління ресурсами та завданнями, і після цього її обробляє інформаційна служба.

Для кожного завдання з черги планувальник визначає відповідний виконавчий комп'ютер та ініціює запуск завдання. Після запуску він веде моніторинг виконання для своєчасного завершення завдання. Виконавчий комп'ютер вважається придатним, якщо він задовольняє вимоги завдання (архітектура, операційна система, дисковий простір, оперативна пам'ять) та очікуване отримання ресурсів забезпечує вчасне виконання, а користувач має права на запуск на цьому комп'ютері.

Особливості невідчужуваних ресурсів обмежують способи розподілу завдань. В гріді застосовують два методи: диспетчер самостійно запускає завдання (push), або виконавчий комп'ютер вимагає завдання від диспетчера (pull). Перший метод широко застосовується у гріді з кластеризованими ресурсами, а для некластеризованих ресурсів ефективніше використовувати другий метод (pull), коли агент на виконавчому комп'ютері повідомляє диспетчеру про готовність прийняти завдання.

Для планування за умов невідчужуваних ресурсів потрібно прогнозувати доступність ресурсів на кожному комп'ютері Grid. Прогноз ґрунтується на даних про те, як комп'ютери віддають процесорний час завданням Grid. Ця інформація збирається на виконавчих комп'ютерах та періодично передається в інформаційну базу. Оскільки виділення часу має випадковий характер, потрібно накопичувати статистику за різними періодами, ураховуючи, наприклад, нічний час та вихідні, коли обчислювальні ресурси часто вільні або слабо використовуються.

Під час планування порівнюють дані про надані ресурси з характеристиками завдання: необхідною кількістю процесорного часу, яку оцінив користувач (walltime), та дедлайном для завершення завдання.

Інформаційна служба проводить двоетапну обробку, спочатку перетворюючи первинну інформацію про ресурси в статистику за різні інтервали часу, наприклад, добу чи тиждень, для відображення кількості ресурсів, які виділяються завданням Grid [35]. Потім проводиться узагальнене представлення загальної кількості ресурсів у кожній групі комп'ютерів, які мають схожі параметри, що дозволяє представити Grid в об'ємній інфраструктурі та публікувати інформацію в зовнішніх системах.

Служба передачі та зберігання даних забезпечує доставку файлів завдань на виконавчі комп'ютери з зовнішніх сховищ та передачу результатів з виконавчих комп'ютерів до зовнішніх сховищ [36]. Для управління завданнями система диспетчеризації GRAM використовує стандартні протоколи доставки стандартних файлів: виконуваного, вхідних даних та діагностики на машину диспетчера.

2.3 Завдання агента на виконавчому комп'ютері

Агентське програмне забезпечення встановлюється на виконавчих комп'ютерах тих користувачів, які бажають надати свої ресурси для використання у Grid. Ресурси виконавчого комп'ютера розподіляються між процесами власника та завданнями Grid. Якщо агентське програмне забезпечення реалізує можливості обмеження використання ресурсів, під час реєстрації виконавчого комп'ютера його власник заповнює конфігураційний файл, де вказує, скільки ресурсів кожного типу (процесор, дисковий простір, оперативна пам'ять і т.д.) він готовий виділяти для Grid інфраструктури. Незалежно від того, чи власник надає такий опис, завдання не може вимагати більше ресурсів, ніж було визначено у його ресурсному запиті, навіть якщо на виконавчому комп'ютері фактично залишаються вільні ресурси. Якщо завдання виявляється надто вимогливим у використанні ресурсів, вказаних у запиті (наприклад, файли завдання займають більше дискового простору або

завдання споживає більше процесорного часу, ніж було заявлено), його виконання припиняється.

Агент відповідає за запуск та управління завданнями на виконавчому комп'ютері, розподіляючи ресурси між процесами власника та завданнями гріду, а також забезпечує передачу даних та безпеку. Безпека тут враховується у трьох аспектах [37, 38]:

– захист власника, тобто забезпечення повної функціональності комп'ютера шляхом контролю рівня споживання ресурсів зовнішнім додатком;

– захист конфігурації комп'ютера, де знаходяться програми та дані;

– захист Grid додатку, включаючи програми, дані та результати.

Агент на виконавчому комп'ютері складається з трьох блоків. Блок запуску завдань повідомляє диспетчеру про вільні ресурси виконавчого комп'ютера та запитує нові завдання. Після цього планувальник вибирає відповідне завдання і повідомляє його ідентифікатор агенту. Потім блок запуску завдань забезпечує доставку файлів завдання на виконавчий комп'ютер, створює виконавче середовище та запускає завдання. Після запуску завдання керування передається блоку розподілу ресурсів.

У випадку, коли агент періодично звертається за новим завданням через визволення ресурсів. Блок розподілу ресурсів забезпечує ізоляцію програм та даних користувача Grid від даних власника комп'ютера, розміщуючи файли завдання в захищеній директорії та виконуючи завдання від імені непривілейованого користувача. Також забезпечує контроль за використанням ресурсів завданням. Інформація про використання ресурсів періодично заноситься до інформаційної бази. Крім того, блок розподілу ресурсів контролює активність власника виконавчого комп'ютера. У випадку, коли завдання перевищило ліміт виділених ресурсів, блок розподілу ресурсів завершує його виконання. Блок управління завданням взаємодіє з операційною системою та блоком розподілу

ресурсів для управління завданням під час його виконання. Якщо завдання не завершено, а власник виконавчого комп'ютера активує свою діяльність, блок управління завданням припиняє його виконання. Після того, як активність власника зменшується, виконання завдання відновлюється.

Отже, обробка завдань на виконавчому комп'ютері відбувається за наступною схемою [39]:

1. Вільні ресурси на виконавчому комп'ютері, які були використані завданням Grid, звільняються, і агент запитує у диспетчера нове завдання:

– блок запуску завдань повідомляє службу взаємодії з агентами, що виконавчий комп'ютер готовий для нового завдання. Ця служба звертається до служби планування, яка видає ідентифікатор найважливішого завдання і змінює статус виконавчого комп'ютера у базі даних;

– потім блок запуску завдань створює на виконавчому комп'ютері необхідну структуру директорій та передає файли завдання через службу передачі та зберігання даних на основі отриманого на попередньому кроці ідентифікатора;

– блок запуску завдань створює виконавче середовище, запускає завдання та передає ідентифікатор системного процесу, що відповідає глобальному завданню, блоку розподілу ресурсів. Крім того, блок запуску завдань через службу взаємодії з агентами змінює статус завдання (виконується);

2. Агент забезпечує виконання завдання на виконавчому комп'ютері з урахуванням використання ресурсів:

– блок розподілу ресурсів контролює кількість спожитих завданням ресурсів, у разі перевищення обсягу ресурсів, зазначеного у запиті, або збільшення навантаження комп'ютера через локальні процеси власника, блок управління завданням відправляє запит на зупинку або призупинення обчислень, також він може зупинити завдання за запитом користувача, у цей час статус завдання змінюється в інформаційній базі;

– під час виконання завдання блок розподілу ресурсів передає диспетчеру інформацію про спожиті ресурси, а блок управління завданням надсилає інформацію про його статус, якщо він змінюється, ці дані також внесені до бази даних;

3. Після завершення обчислень агент завершує процес виконання завдання:

– після завершення завдання блок запуску надсилає вихідні файли через службу передачі та зберігання даних, після цього всі дані, пов'язані із завданням, видаляються з виконавчого комп'ютера;

– служба передачі та зберігання даних повідомляє службу управління завданнями про завершення завдання та змінює його статус в інформаційній базі на «завершено».

Блок запуску завдань перевіряє статус виконавчого комп'ютера у блоку розподілу ресурсів. Якщо ресурси вільні, блок запуску завдань повідомляє диспетчер, що виконавчий комп'ютер готовий для нового завдання, і цикл починається знову.

2.4 Функції користувальницького інтерфейсу

Інтерфейс користувача – це клієнтська програма, створена з використанням API служб диспетчера [40]. Ця програма дозволяє користувачеві надсилати файл із завданням у мові RSL та запускати його в фоновому режимі, отримувати інформацію про його стан та управляти ним.

Інтерфейс користувача забезпечує три основні функції [41]:

– введення нового завдання;

– скасування введеного завдання;

– отримання інформації про статус та хід виконання введеного завдання.

Під час виконання завдання в пакетному режимі, користувач має змогу слідкувати за станом завдання, переглядати кількість використаних ресурсів та при необхідності зупиняти виконання завдання.

Взаємодія користувача з клієнтською програмою відбувається через інтерфейс командного рядка. Звернемо увагу, що в процесі розробки користувацького додатка зберігається набір операцій та їх назви, які відповідають клієнтській програмі служби GRAM у складі інструментарію Globus Toolkit, а саме утиліті globusrun-ws [42]. Визначення параметрів кожної операції також проводиться за допомогою параметрів командного рядка, аналогічних до globusrun-ws.

Для того, щоб створити нове завдання, користувач повинен підготувати опис завдання мовою RSL. Формат, що використовується у розробленій системі, ґрунтується на форматі опису завдань, який підтримується службою GRAM, і також містить деякі параметри, необхідні для забезпечення планування у однорівневій Grid системі.

Введення завдання здійснюється через виконання команди:

$ sard_run –submit –f <job_description_file> [–c <proxy_cert>] <service>

У цьому випадку програма виконує операцію введення завдання (-submit), звертаючись до служби управління завданнями за її URL <service>, і отримує на вхід файл опису завдання <job_description_file>. Необов'язковий параметр «-c» дозволяє вказати директорію розміщення проксі-сертифіката користувача, відмінну від директорії за замовчуванням.

Після синтаксичної перевірки опису завдання клієнтська програма делегує повноваження користувача системі диспетчеризації за допомогою служби делегування зі складу Globus Toolkit (Delegation Service [43]). Делегування повноважень є стандартним механізмом, який використовується для того, щоб диспетчер міг подальше доставляти вхідні та результуючі файли з правами

користувача. Якщо опис складено коректно, реєструється завдання в системі, після чого користувачу повертається унікальний ідентифікатор, який він може використовувати для управління завданням. Для перевірки опису без фактичного введення завдання можна використати параметр «-validate», що полегшує пошук помилок у описі. Для цього необхідно запустити програму так:

$ sard_run –validate –f < job_description_file>

Управління завданням відбувається за допомогою операцій –kill та –status , які дозволяють відповідно скасувати виконання завдання чи отримати інформацію про його стан. Скасування завдання можна виконати, використовуючи параметр «-i», який вказує унікальний ідентифікатор завдання:

$ sard_run –kill –i <job_id> [–c <proxy_cert>] <service>

Після успішного скасування завдання воно переходить у кінцевий стан KILLED. Операція отримання статусу завдання (-status) дозволяє користувачеві отримати інформацію про стан завдання, а також отримати додаткову інформацію, наприклад, час введення завдання, час запуску завдання на виконавчому комп'ютері, астрономічний та процесорний час виконання завдання, а також повідомлення про останню помилку, якщо така виникла.

2.5 Обґрунтування вибору Grid систем з ієрархічним управлінням

Система XtremWeb дозволяє створювати обчислювальні мережі, де кожен комп'ютер може виконувати одну з трьох ролей: координатора, виконавця або клієнта. У мережі може бути лише один координатор, а кількість виконавців та клієнтів необмежена.

Координатор відповідає за керування завданнями у системі XtremWeb, розподіляє їх та зберігає результати обчислень. Ця система ґрунтується на pull-моделі взаємодії між виконавцями та координатором. Такий підхід дозволяє

вирішувати проблеми взаємодії з комп'ютерами у локальних мережах або захищеними мережевими екранами.

XtremWeb підтримує обробку різноманітних додатків – від скомпільованих до Java-додатків. Планування відбувається за принципом черги (FIFO – first in, first out). Виконавець обирає наступне завдання з черги на основі вимог до ресурсів, таких як тип процесора та версія операційної системи.

Після запуску завдання координатор очікує результат виконання. У випадку помилки завдання перепланується та запускається на іншому комп'ютері. Клієнти системи дають користувачам можливість запускати завдання та отримувати результати. Користувач надає координатору необхідні файли та параметри для запуску.

Виконавці – це компоненти, що встановлюються на виконавчі комп'ютери. Вони завантажують та виконують завдання, періодично надсилаючи сигнали координатору про стан обробки. У разі помилки завдання перезапускається.

Система XtremWeb може керувати обчислювальною інфраструктурою з великою кількістю комп'ютерів. Для оптимізації можна використовувати різні методи, такі як розподіл навантаження, використання іншого програмного забезпечення для балансування та інші. Однак реалізація таких рішень лежить на адміністраторах конкретної інфраструктури.

Однією з популярних систем у сфері розподіленого обчислення є Condor [24], розроблена університетом Вісконсіна, США. Ця система об'єднує комп'ютери для вирішення складних завдань, розподіляючи їх обчислювальні потужності, навіть у моменти простою. На відміну від традиційних систем, Condor пропонує розподіл завдань як на відчужені, так і на не відчужені комп'ютери.

Ця система концентрується на об'єднанні ресурсів у пулі, що дозволяє розподіляти завдання на виконавчі комп'ютери, керувати їх виконанням і ресурсами. Розподіл ресурсів базується на мові ClassAd (Classified Advertisement), що дозволяє описувати комп'ютерні ресурси та їх вимоги. Condor забезпечує пошук виконавчих комп'ютерів для кожного завдання, спираючись на інформацію про комп'ютери та ресурсні запити.

Компоненти Condor на виконавчих комп'ютерах дозволяють поділ ресурсів між локальними та зовнішніми завданнями, забезпечуючи при цьому невідчужуваність комп'ютера. Підвищення активності власника може призупинити виконання Condor-завдання, але його можна буде відновити з місця призупинення.

Для початку роботи з Condor, користувачу необхідно створити файл опису завдання, вказати вимоги до виконавчих комп'ютерів та підготувати необхідні файли на машині запуску. Ця інформація створює опис завдання, що використовується для пошуку ресурсів і додавання завдання до черги.

У Condor підтримується розподілена черга завдань на двох рівнях: локально на кожній машині запуску і в глобальній черзі на центральному менеджері. Завдання виконуються з урахуванням пріоритетів користувачів.

Після знаходження ресурсів для завдання, воно запускається на виконавчому комп'ютері, а машина запуску відстежує його виконання. Ці процеси контролю можна призупинити або перевіряти статус завдання. Після завершення завдання результати автоматично передаються на машину запуску.

Condor також підтримує механізми міграції завдань, контрольних точок та віддаленого виклику процедур, що дозволяють ефективно виконувати завдання на ресурсах пулу. Забезпечені механізми безпеки та цілісності даних гарантують надійну передачу інформації між компонентами та захист даних користувача і власника ресурсу.

Особливий тип розподілених комп'ютерних систем – це корпоративні рішення, спрямовані на оптимізацію роботи парку комп'ютерів підприємства. Ці розробки використовують комп'ютери організації для вирішення завдань, що потребують великих ресурсів (наприклад, аналіз фінансових ризиків), коли вони не надто використовуються або знаходяться в стані простою. Наприклад, у системі Entropia [23] впроваджено новий підхід до об'єднання персональних комп'ютерів для обробки завдань, включаючи послідовні операції.

Структура Entropia включає сервер та безліч клієнтів, на яких функціонує клієнтська частина системи. Сервер розбиває завдання на безліч підзавдань та

запускає їх на клієнтських машинах. Після завершення обчислень результати підзадач об'єднуються та повертаються користувачеві.

Програмна інфраструктура системи Entropia складається з трьох рівнів. Компоненти управління виконавчими комп'ютерами та розподіл ресурсів встановлені на клієнтських машинах та сервері відповідно. Компоненти керування завданнями розташовані лише на сервері. Разом із системою Entropia можуть працювати інші системи управління завданнями. Ця система забезпечує виконання будь-якого Win32-додатка без будь-яких змін, використовуючи його у віртуальному середовищі.

2.6 Обґрунтування використання систем Condor та Globus Toolkit

У рамках реалізації Grid системи з кластеризованими ресурсами, система Condor виконує ряд функцій управління завданнями на виконавчому комп'ютері: розподіл внутрішніх ресурсів комп'ютера, перенесення всіх необхідних файлів з сервера диспетчера на виконавчий комп'ютер та передачу результатів виконання завдань з виконавчого комп'ютера на сервер диспетчера. Для використання цих можливостей системи Condor необхідно встановлювати компоненти центрального керування та машини запуску на сервері диспетчера, а на виконавчих комп'ютерах – компоненти виконавчої машини Condor. Структура системи диспетчеризації з компонентами Condor представлена на рисунку 2.2.

Вибір програмного інструментарію Globus Toolkit був обумовлений кількома моментами. По-перше, використання цього інструментарію дозволяє створити систему, яка взаємодіє з наявними розробками, що використовують GT. По-друге, він надає можливість використовувати готові та опрацьовані служби базового рівня, які, зокрема, забезпечують безпеку. І нарешті, для роботи служб диспетчера Grid системи використовувався компонент GT. У наступній частині роботи розглянемо всі три архітектурні компоненти системи диспетчеризації та їх функції.

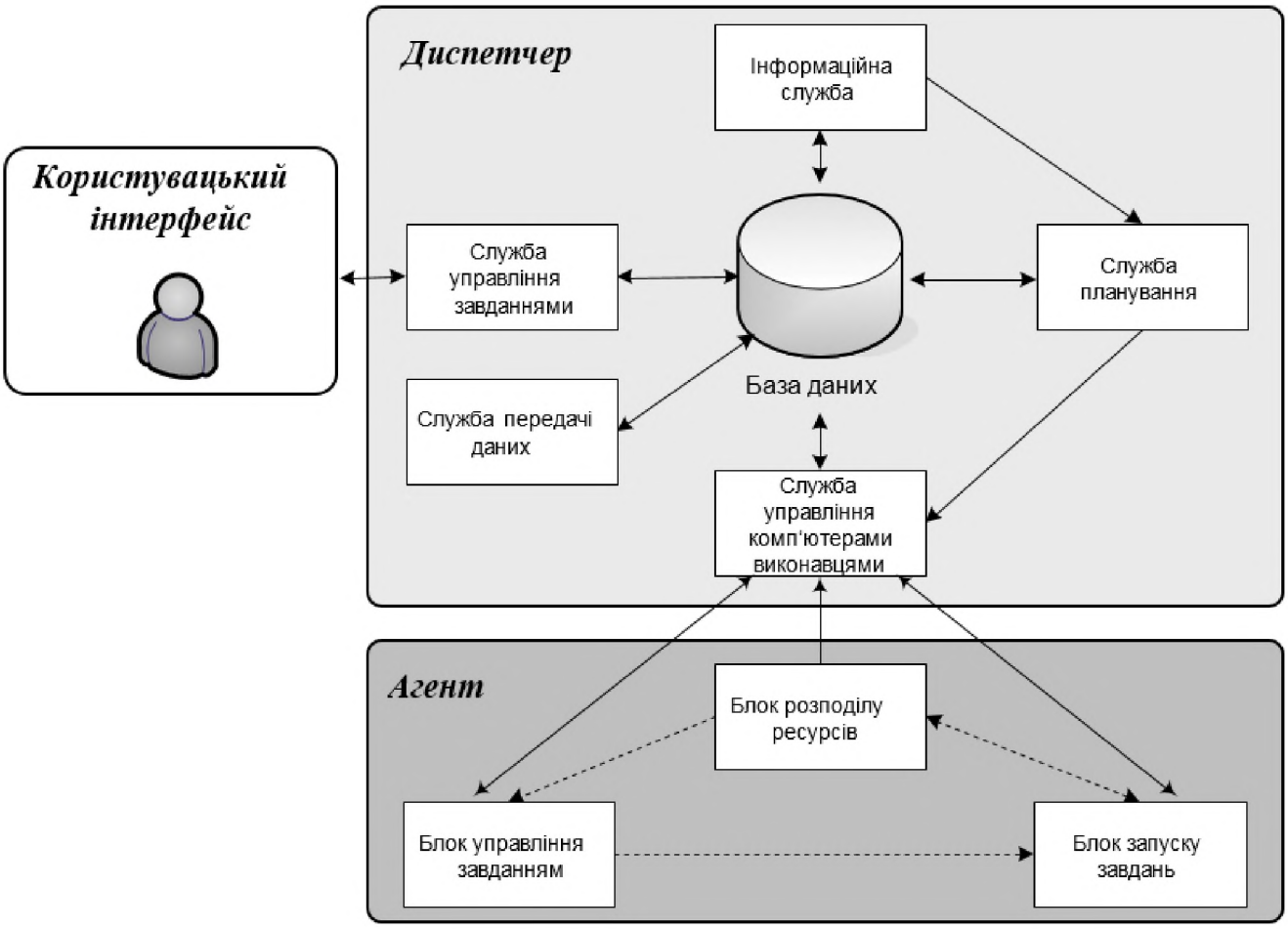

Рисунок 2.2 – Архітектура Grid з інтеграцією компонентів системи Condor

Диспетчер однорівневого Grid представлений набором вебслужб, що працюють у контейнері Globus Toolkit. Цей розділ присвячений аналізу деталей реалізації служб, що входять до складу диспетчера Grid системи, їх функцій та механізмів взаємодії між собою та іншими компонентами системи.

Для зберігання інформації про користувачів, завдання, виконавчі комп'ютери та необхідну службову інформацію використовується база даних, яка працює під управлінням СУБД PostgreSQL [41].

Служба взаємодії з агентами організовує взаємодію агента з диспетчером за допомогою наступних повідомлень: реєстрація, зняття з реєстрації, підключення, періодичний звіт та вимкнення. Розглянемо обробку кожного з цих типів повідомлень:

1. Реєстрація виконавчого комп'ютера: При повторному підключенні виконавчого комп'ютера до інфраструктури агент надсилає диспетчеру відповідне повідомлення, яке містить інформацію про продуктивність комп'ютера. Після отримання цього повідомлення диспетчер заносить отриману інформацію до бази даних та повертає агенту код результату та ідентифікатор, присвоєний новому комп'ютеру. Якщо код результату свідчить про успішну реєстрацію комп'ютера, агент записує ідентифікатор у файл та використовує його при подальшій взаємодії зі службою.

2. Зняття комп'ютера із реєстрації: У разі вирішення власником виключити свій комп'ютер з інфраструктури, він виконує команду зняття виконавчого комп'ютера з реєстрації. Під час обробки запиту диспетчер видаляє запис про комп'ютер з бази даних і повертає агенту код результату.

3. Підключення комп'ютера: Отримавши це повідомлення, диспетчер знає про старт агента на зареєстрованому комп'ютері. Перед надсиланням цього повідомлення агент збирає інформацію про характеристики комп'ютера та надсилає диспетчеру. Диспетчер оновлює інформацію про комп'ютер у базі даних та вважає його доступним. У відповідь служба повідомляє агенту інтервал часу, через який він має надіслати черговий звіт.

4. Періодичний звіт: Отримавши від диспетчера інтервал часу для надсилання чергового звіту, агент починає збирати інформацію про функціонування комп'ютера. До складу звіту входить ідентифікатор комп'ютера, описаний інтервал часу та інформація про частку вільного процесорного часу за вказаний інтервал (середнє значення). Якщо на комп'ютері виконуються завдання Grid, до звіту також включається його стан, час виконання на комп'ютері та кількість спожитого завданням процесорного часу. Завдання Grid може перебувати в одному з чотирьох станів: запущено, виконується, завершено або аварійно завершено. У відповідь агенту повертається інтервал до наступного звіту.

5. Вимкнення виконавчого комп'ютера: Після штатного завершення агента він надсилає диспетчеру повідомлення про відключення виконавчого комп'ютера зі звітом про функціонування комп'ютера за інтервал часу з моменту надсилання останнього звіту. Отримавши таке повідомлення, диспетчер вважає відповідний комп'ютер недоступним, а завдання Grid, якщо такі існують, що виконуються на ньому, аварійно завершилися. У разі успішної обробки цього повідомлення агент отримує відповідь, яка не містить інформації.

Варто також враховувати, що під час роботи системи можуть виникати різноманітні виняткові ситуації, які потребують відповідної обробки для забезпечення консистентності системи. Серед таких ситуацій можуть бути:

- недоступність диспетчера при запуску агента;
- помилки у роботі агента, через які він перестає надсилати періодичні звіти.;
- проблеми мережі між диспетчером та агентом, коли диспетчер перестає отримувати звіти від агента;
- перезавантаження диспетчера, яке може призвести до виникнення проблем.

Під час обробки завдань може виникати до дванадцяти різних станів Grid системи (рис. 2.3). При створенні нового завдання користувач готує його опис у форматі XML і передає його службі через інтерфейс користувача. Якщо опис вірний, з нього видобувається необхідна інформація, яка заноситься у відповідні таблиці бази даних диспетчера. Завдання отримує статус SUBMITTED, а користувач отримує унікальний ідентифікатор завдання. У випадку невірного опису інформація не потрапляє до системи, а користувач отримує повідомлення про помилку. Після успішного створення завдання починається процес передачі необхідних вхідних даних із зовнішніх сховищ на комп'ютер диспетчера, про що свідчить статус STAGEIN.

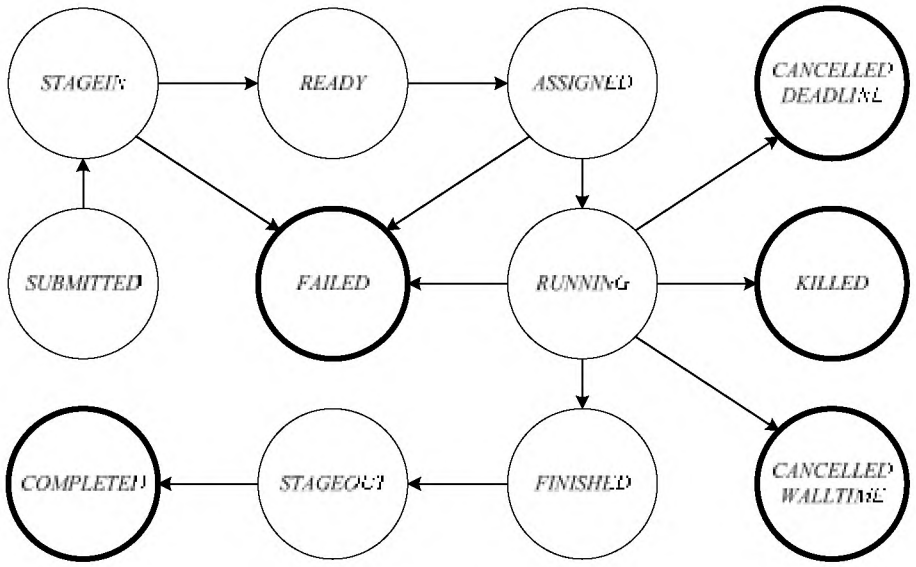

Рисунок 2.3 – Діаграма станів завдання у системі диспетчеризації

Цей процес здійснюється службою передачі файлів, виділений окремо, оскільки може займати тривалий час. Окремий статус допомагає зрозуміти, що саме відбувається з завданням і не починати його планування, доки не буде завантажено весь набір необхідних файлів. Після завершення процесу передачі файлів служба повідомляє службу управління завданнями, яка змінює статус завдання на READY та повідомляє планувальника, що завдання готове до обробки та розподілу на виконавчий комп'ютер. Таким чином, завдання потрапляє у чергу планувальника лише після завершення доставки даних для нього. Якщо в процесі доставки виникають помилки і не всі файли можуть бути доставлені на машину диспетчера, завдання переходить у статус FAILED і його обробка в системі припиняється.

Після успішної доставки усіх необхідних файлів планувальник здійснює пошук найбільш підходящого комп'ютера для виконання завдання. Коли планувальник вирішує запустити завдання на одному з доступних виконавчих комп'ютерів, він викликає відповідний метод служби управління завданнями, яка заносить необхідну інформацію до таблиці призначень та змінює статус завдання на ASSIGNED. Це означає початок запуску завдання, включаючи доставку

необхідних файлів на виконавчий комп'ютер. Після призначення завдання на комп'ютер, він вважається зайнятим і не розглядається для подальшого планування.

Варто зауважити, що присутність цього стану є надзвичайно важливою в умовах використання системи Condor (або будь-якої іншої зовнішньої системи), оскільки диспетчер фактично запускає завдання в цю систему, і процес запуску може затягнутися з невідомих для диспетчера причин. Зокрема, завдання можуть «зависати» в черзі Condor, тому такі ситуації повинні бути коректно оброблені диспетчером. При виявленні випадків затримки завдання знімається з виконання та повертається у чергу для подальшого планування.

Згідно призначень, завдання запускається на виконання за допомогою системи Condor. Прямий запуск завдання на виконавчому комп'ютері здійснюється через програму condor_submit, включену в програмне забезпечення Condor. Ця програма отримує на вхід опис завдання у спеціальному форматі, який генерується на основі даних таблиці завдань та таблиці передачі даних. У разі успішного запуску condor_submit повертає внутрішній ідентифікатор завдання у системі Condor, який зберігається в таблиці призначень та використовується пізніше для порівняння з унікальним ідентифікатором завдання в системі диспетчеризації. Важливо зауважити, що за замовчуванням Condor самостійно шукає відповідний комп'ютер для запуску, тому щоб запустити завдання на тому комп'ютері, що був визначений планувальником системи диспетчеризації, необхідно явно вказати це в описі завдання. Condor забезпечує таку можливість за допомогою додавання до опису завдання спеціальної вимоги Requirements = Machine==<hostname>, де <hostname> – ім'я виконавчого комп'ютера, призначеного для виконання завдання планувальником.

Після запуску завдання через condor_submit, воно потрапляє у чергу системи Condor. Проте з різних причин, навіть коли комп'ютер готовий прийняти нове

завдання, воно може простоювати у черзі. Оскільки така поведінка є характерною для системи Condor, служба взаємодії з агентами сповіщає про безпосередній старт завдання на виконавчому комп'ютері службі управління та протягом усього часу його виконання регулярно передає отриману від агента інформацію. Це призводить до переходу завдання у стан RUNNING.

Якщо під час запуску або виконання завдання на комп'ютері виникнуть помилки, воно отримає статус FAILED. Якщо виконавчий комп'ютер з якоїсь причини відключиться від системи диспетчеризації під час виконання завдання, останнє буде перезапущено на іншому комп'ютері. У такому випадку завдання повернеться до статусу READY і буде повторно оброблено планувальником.

Коли служба взаємодії з агентами повідомить про завершення виконання завдання на виконавчому комп'ютері, воно буде включено до списку завдань, які очікують завершення обробки системою Condor. Після завершення виконання завдання може пройти певний час, поки Condor звільнить цей комп'ютер. Тому служба керування завданнями періодично запитує статус завдань Condor через condor_status. Як тільки Condor підтвердить завершення завдання, воно отримає статус FINISHED.

Коли завдання переходить у стан FINISHED, виконавчий комп'ютер звільняється, а саме завдання отримує статус STAGEOUT. Під час цього стану відбувається передача результатів на зовнішні сховища, якщо це було вказано користувачем у вихідних даних завдання. Остаточний стан, позначений жирною лінією на діаграмі, COMPLETED, що означає успішну передачу результатів і завершення обробки завдання.

Інші кінцеві стани включають FAILED, який виникає при помилках під час планування, виконання або доставки файлів, а також KILLED, CANCELLED_WALLTIME і CANCELLED_DEADLINE. Останні два стани вказують на примусове скасування виконання завдання диспетчером через

перевищення максимального часу (maxWallTime) та крайнього терміну (deadline) відповідно. Якщо ці параметри перевищені, планувальник скасовує виконання завдання, яке отримує статус KILLED. Такий самий статус присвоюється завданню, якщо його скасує користувач.

2.7 Обґрунтування вибору архітектури агента на виконавчому комп'ютері

Комплект агентського програмного забезпечення включає в себе агента, компонент системи Condor, що встановлюється на виконавчій машині, а також спеціальну утиліту – лідера, яка передає агенту інформацію про фактичний запуск завдання. Засоби Condor дозволяють замість запуску завдання виконувати програму, вказану в конфігураційному файлі виконавчої машини. Ця можливість використовується в SARD для того, щоб агент отримав інформацію про завдання Grid. Коли компонент системи Condor отримує нове завдання, він запускає програму-лідера, яка в свою чергу запускає завдання. Лідер повідомляє агенту ідентифікатор процесу, що запустився, і ідентифікатор завдання в базі даних диспетчера.

Програмне забезпечення Condor на виконавчому комп'ютері поділене на кілька демонів, список яких знаходиться у конфігураційному файлі і може бути розширено. При старті системи Condor стартує головний демон системи – master, який запускає інші демони і відповідає за їх постійну роботу. У випадку, якщо якийсь з демонів перестає працювати, master-демон автоматично його перезапускає. Цей механізм використовується для одночасного запуску та завершення Condor та агента. Для цього агент був розроблений як демон системи Condor та доданий до списку демонів у конфігураційному файлі.

Управління завданнями на виконавчому комп'ютері та передача вхідних і результуючих файлів між машиною диспетчера і виконавчим комп'ютером у SARD покладено на компоненти системи Condor. Крім цього, в агенті реалізовані функції реєстрації (і зняття з реєстрації) виконавчих комп'ютерів у системі диспетчеризації Grid системи з кластеризованими ресурсами, збору інформації про споживання ресурсів комп'ютера, а також надсилання періодичних звітів із зібраною інформацією, повідомленнями про функціонування самого агента та подіями, пов'язаними зі зміною статусу завдання Grid.

Агент має модульну архітектуру, що складається з трьох шарів: ядра, шару взаємодії із системою та оболонки агента. Такий підхід дозволяє ефективно розділити кросплатформні та системно-залежні компоненти агента, що спрощує створення версій агента для різних платформ та його модернізацію. Крім того, архітектура агента уможливлює заміну запозичених компонентів на власні в майбутньому.

Основою агента є кросплатформне ядро. Метою цього ядра є обробка подій, таких як ініціалізація, реєстрація та зняття з реєстрації комп'ютера, підключення, відключення, оновлення стану та запуск нового завдання. Метод ініціалізації викликається при старті агента і призводить до його налаштування та готовності до роботи.

Події реєстрації, зняття з реєстрації, підключення та відключення спричиняють відправку відповідних запитів до диспетчера (якщо це можливо в поточному стані агента) і при отриманні відповідей, що підтверджують успішне виконання операцій, змінюють статус агента (зареєстрований/незареєстрований, підключений/відключений).

Оболонка агента відповідає за генерацію подій під час роботи агента, які потім обробляються ядром. Крім того, оболонка створює екземпляри конкретних системно-залежних компонентів та передає їх у ядро.

Шар взаємодії із системою виконує такі функції як зберігання інформації про конфігурацію агента, збір інформації про характеристики виконавчого комп'ютера, виконання тесту продуктивності для визначення показників комп'ютера та стеження за завданнями Grid.

Отже, ця архітектура агента уможливлює його ефективну роботу, розділяючи функціонал на різні компоненти та забезпечуючи гнучкість при модифікації та оновленні.

При виклику методів ядра, оболонка ініціює різноманітні типи подій, про які вже було згадано раніше. Реєстрація та видалення комп'ютера з реєстрації відбуваються під час запуску агента з вказаними параметрами у командному рядку. При звичайному запуску агента на зареєстрованому комп'ютері створюється підключення до диспетчера. У випадку невдачі (наприклад, через відсутність диспетчера), оболонка розпочне повторні спроби через певні інтервали часу. Під час виконання регулярно відбувається оновлення стану. Коли агент отримує від засобів запуску повідомлення про запуск нового завдання, оболонка передає його ядру.

Коли операційна система вимагає завершення програми, агент передає запит диспетчеру для відключення. Розглянемо докладніше особливості реалізації агентської частини системи диспетчеризації. Конфігураційна інформація зберігається у файлі agent.config, який читається при запуску агента. Найважливішим та обов'язковим параметром у цьому файлі є адреса служби диспетчера для керування агентами.

Після успішної реєстрації виконавчого комп'ютера агент записує присвоєний йому ідентифікатор у файл. При наступних запусках агент намагатиметься зчитати цей файл. Якщо ідентифікатор успішно зчитується, виконавчий комп'ютер вважатиметься зареєстрованим. Якщо файл відсутній, комп'ютер вважатиметься незареєстрованим, і робота агента буде неможлива.

Для реєстрації комп'ютера у системі необхідно запустити агента зі спеціальним параметром командного рядка «-register». Щоб видалити виконавчий комп'ютер з реєстрації, агент повинен бути запущений з параметром «-unregister», що призведе до передачі диспетчеру запиту на видалення з реєстрації. Після успішного видалення з реєстрації файл, що містить ідентифікатор комп'ютера, буде видалено.

Для отримання інформації про характеристики комп'ютера використовуються стандартні функції операційної системи. На платформі Windows у реалізації агента використовуються функції GetSystemInfo, GetComputerName, GetVersionEx та GlobalMemoryStatus. Результати цих функцій перетворюються відповідно до внутрішнього формату зберігання і передаються ядру агента для надсилання диспетчеру.

Компонент агента, що відповідає за отримання інформації про продуктивність процесора, реалізований з урахуванням відомого тесту LINPACK [39]. Цей тест полягає у вирішенні системи лінійних рівнянь методом Гауса з матрицею заданого розміру, заповненою випадково згенерованими коефіцієнтами. Результат ділиться на час виконання тесту, що дає продуктивність процесора в мільйонах операцій з плаваючою точкою в секунду (Mflops). При даному методі виміру враховується можлива похибка, пов'язана з неповним використанням процесора при виконанні тесту, адже вимірюється витрачений процесорний час, а не загальний час тесту.

Головною функцією агента в контексті розподілу ресурсів виконавчого комп'ютера є збір інформації про завдання Grid. Для цього важливо відстежувати дочірні процеси, що виникають під час виконання основного процесу завдання, та обмежувати кількість споживаних ним ресурсів.

Для вирішення цих завдань на виконавчих комп'ютерах під управлінням ОС Windows використовується API Job Object [20], яке надає засоби для відстеження

виконання групи процесів. Об'єкт ядра – завдання (job) – був введений в ОС Windows. Використання об'єктів-завдань дозволяє накладати обмеження на сукупність процесів, наприклад, обмеження обсягу оперативної пам'яті та часу процесора. Також вони забезпечують захист процесів та даних від несанкціонованого доступу до захищених ресурсів.

Під час створення екземпляра об'єкта-завдання агент зупиняє його процес, щоб уникнути порушення обмежень. При цьому йде спостереження за всіма процесами, що породжуються, щоб не втратити контроль над ними, навіть якщо батьківський процес завершиться раніше дочірнього.

Під час виконання завдання Grid ядро агента періодично звертається до компонента, який відслідковує процеси, щоб оновити інформацію про об'єкт-завдання. Ця інформація включає загальний час, присвячений процесору, та кількість активних процесів у цьому об'єкті. На підставі цієї інформації обчислюється загальний час, витрачений на роботу завдання, і його поточний статус. Завдання Grid вважається завершеним, якщо в об'єкті-завданні не лишиться жодного активного процесу.

Агент взаємодіє з диспетчером, надсилаючи повідомлення відповідній вебслужбі диспетчера та отримуючи відповіді. Формат передаваних повідомлень визначається протоколом SOAP [21]. На стороні вебслужби підтримка SOAP реалізована в інструментарії Globus Toolkit. Для підтримки цього протоколу з боку агента застосовується кросплатформова бібліотека gSOAP [22].

Для інтеграції з диспетчером використовується опис вебслужби диспетчера WSDL. Цей опис використовується програмою кодогенерації, яка входить до складу бібліотеки gSOAP, для генерації коду мовою C++, який реалізує взаємодію зі службою.

Висновки до розділу 2

У другому розділі описується архітектура системи управління завданнями для Grid системи з кластеризованими ресурсами. Склад, структура та функції компонентів системи відповідають вимогам, викладеним у першому розділі. Архітектура складається з трьох компонентів: диспетчера, агента та інтерфейсу користувача. Диспетчер реалізується через набір вебслужб та розташовується на виділеному комп'ютері у Grid сегменті. Ресурси виконавчих комп'ютерів Grid інфраструктури не доступні користувачам безпосередньо, оскільки всі зовнішні інтерфейси доступу до них централізовані в диспетчері. Основна функція агента, що встановлюється на виконавчому комп'ютері, полягає у керуванні завданням на етапі виконання. Інтерфейс користувача дозволяє керувати власними завданнями у стандартному для Grid форматі та отримувати інформацію про їхній стан.

РОЗДІЛ 3
МОДЕЛЬ ФУНКЦІОНУВАННЯ GRID СИСТЕМИ З КЛАСТЕРИЗОВАНИМИ РЕСУРСАМИ ДЛЯ ПЛАНУВАННЯ ЗАВДАНЬ

3.1 Модель архітектури Grid системи з кластеризованими ресурсами

Зважаючи на складність завдань організації надання ресурсів у Grid середовищі, моделювання та вимірювання їх якості обслуговування стає нетривіальним завданням. У даному випадку, розглядатимемо ці процеси поступово, від простого до складного, щоб накопичувати знання поетапно. На рис. 3.1 зображено модель архітектури Grid системи з кластеризованими ресурсами [8].

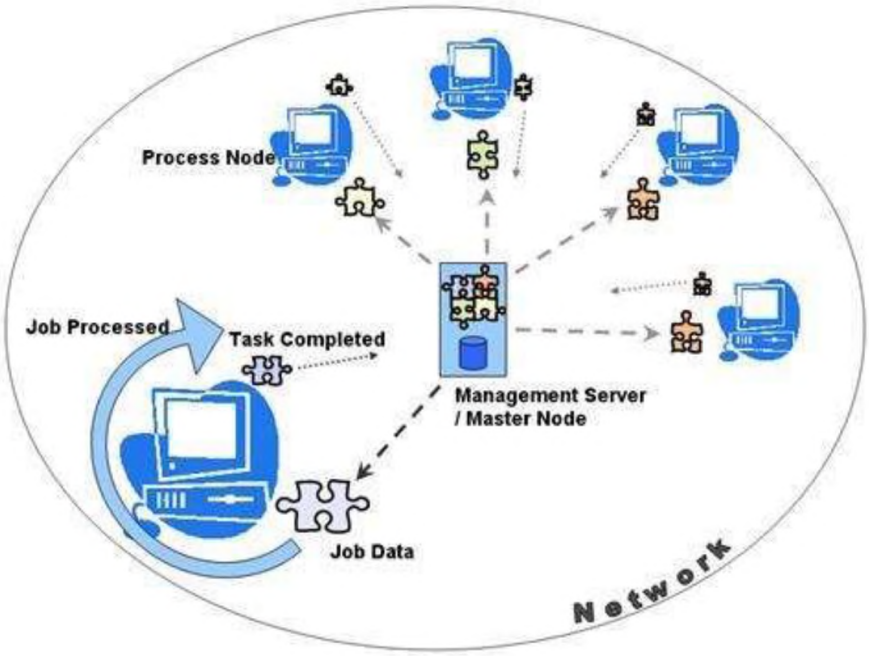

Рисунок 3.1 – Архітектура Grid системи з кластеризованими ресурсами

Після надходження заявки на обслуговування, вона потрапляє у чергу (Management Server / Master Node, MN). В MN можливі два види відмов в обслуговуванні заявки – при переповнюванні черги вхідних заявок системи менеджера і при недостатності фізичних, віртуальних і буферних ресурсів безпосередньої ланки обслуговування заявки.

3.2 Вибір показників якості обслуговування, доступності і енергоефективності функціонування Grid системи

Згідно з джерелами [26, 29], у ході виконання Grid обчислень беруть участь сторони споживача і провайдера послуги, а також кілька посередників. Між ними укладається угода про якість послуг (SLA), яка регулює параметри якості, що надаються провайдером. Ці параметри встановлюються при умові, що споживач дотримується вхідних параметрів інтенсивності звернень для надання послуг. Угода SLA визначає такі параметри якості:

– затримка відгуку хмарного сервера;

– доступність хмарного сервера.

Доступність є комплексним показником, який обумовлений зайнятістю ресурсів, їх надійністю та безпекою. Зазвичай, провайдер не розголошує причини простою та недоступності обладнання, а користувач оцінює загальний час доступності серверів, не деталізуючи їх функціонування. Однак це не означає, що причини простоїв можна ігнорувати при оцінці параметрів доступності, якщо вони не включені у відповідні пункти угоди SLA.

Крім виконання пунктів угоди SLA, провайдери послуг використовують додаткові критерії для оцінювання продуктивності хмарної інфраструктури. Основні з них це мінімізація витрат енергії обладнанням обчислювальної інфраструктури; а також оптимізація витрат на обслуговування обчислювальної інфраструктури, включаючи енергію, підтримку високої доступності та безпеки; а також розробку тарифних планів послуг з метою максимізації прибутку.

3.3 Побудова моделі менеджера Grid системи з кластеризованими ресурсами

Ця модель заснована на використанні апарату моделювання Continuous Time Markov Chain (CTMC) і показана на рис. 3.3.

Модель є варіацією системи масового обслуговування (СМО) типу M/M/1/N, де:

M – марковський (з експоненціальним розподілом і інтенсивністю λ) потік вхідних заявок;

M – марковські (з експоненціальним розподілом і інтенсивностями обслуговування δh, δw, δc) потоки обслуговування заявок – пошуку ресурсів в гарячому/теплому/холодному пулах відповідно;

1 – кількість обслуговуючих пристроїв (один сервер MN);

N – розмір черги (буфера) заявок.

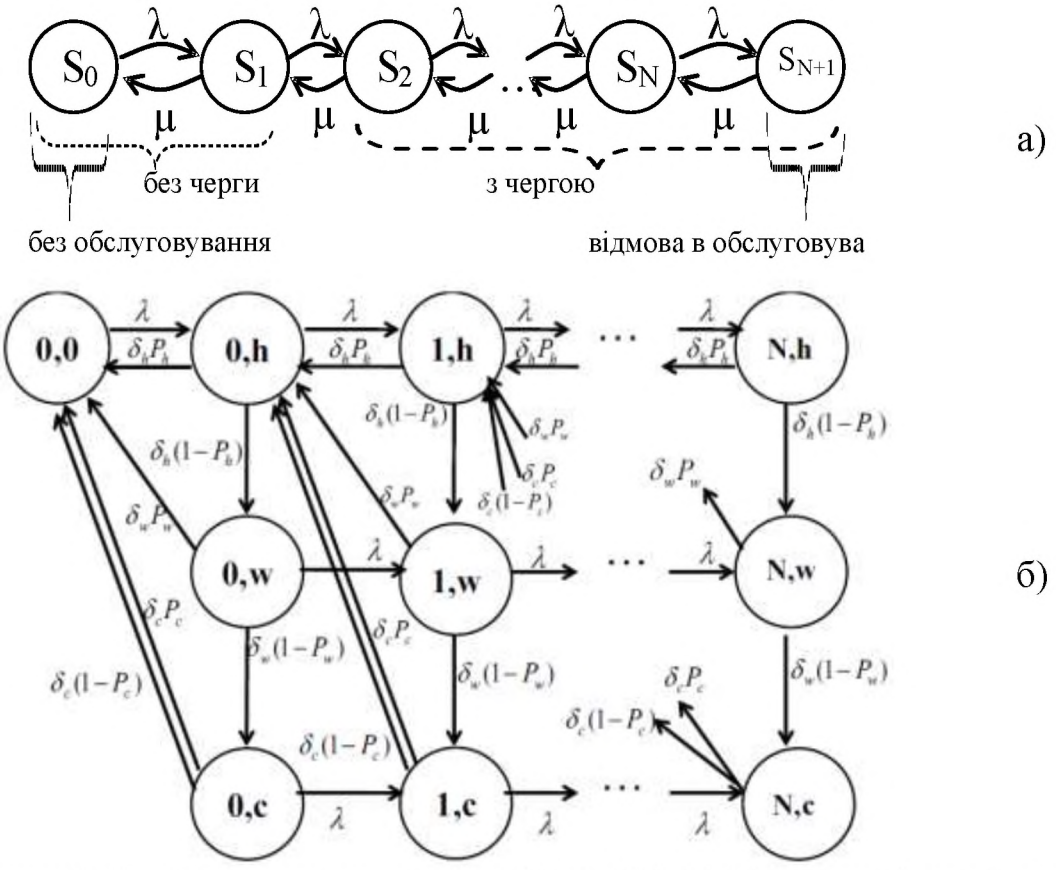

Рисунок 3.3 – Класична модель СМО M/M/1/N (а) і модель функціонування менеджера MN (б), для стану $S_{i,j}$: i – кількість заявок в черзі, j – тип пулу, [h, w, c]

Менеджер Grid системи функціонує таким чином. Після надходження першої заявки система переходить в стан пошуку вільного ресурсу в гарячому пулі серверів ($S_{0,h}$). З урахуванням вірогідності обслуговування заявки в гарячому пулі P_h, менеджер або знайде необхідний ресурс і повернеться в початковий стан $S_{0,0}$ з

інтенсивністю $P_h\delta_h$, або перейде з інтенсивністю $(1 – P_h)\delta_h$ в стан пошуку ресурсу в теплому пулі ($S_{0,\ w}$). Аналогічно, з інтенсивністю $P_w\delta_w$, менеджер може знайти вільний ресурс в теплому пулі і повернутися в початковий стан, а може і не знайти його і з інтенсивністю $(1 – P_w)\delta_w$ перейти в стан пошуку ресурсу в холодному пулі ($S_{0,\ c}$). Із стану $S_{0,c}$ менеджер повертається в початковий стан і при знаходженні вільного ресурсу в холодному пулі (з інтенсивністю $P_c\delta_c$) так і за відсутності вільного ресурсу (з інтенсивністю $(1 – P_c)\delta_c$), але в останньому випадку заявка буде не обслужена через нестачу ресурсів в Grid.

Як і в класичній моделі СМО, коли менеджер Grid здійснює пошук вільних ресурсів (стани $S_{0,h}$, $S_{0,w}$ і $S_{0,c}$), на його вхід може поступити нова заявка з інтенсивністю λ. Ця заявка буде поміщена в чергу (буфер) і система відповідно перейде в стани $S_{1,h}$, $S_{1,w}$ і $S_{1,c}$. Процес обслуговування (пошуку ресурсу) в цих станах нічим не відрізняється від процесу, описаного вище.

Grid менеджер має обмежену довжину черги заявок, тому, досягнувши станів $S_{N,\ h}$, $S_{N,\ w}$ і $S_{N,\ c}$ наступні заявки отримають відмову в обслуговуванні, обумовлену обмеженою довжиною черги/буфера MN.

Вхідними параметрами моделі є:

λ – інтенсивність вхідного потоку заявок на обслуговування;

$1/\delta_h$, $1/\delta_w$, $1/\delta_c$ – середній час пошуку вільних ресурсів в гарячому/теплом/холодному пулах серверів відповідно;

P_h, P_w, P_c – вірогідність успішного обслуговування заявки в гарячому/теплому/холодному пулах серверів відповідно.

Кількісні значення вхідних параметрів представлені в табл. 3.1.

Таблиця 3.1 – Значення вхідних параметрів моделі менеджеру MN

Вхідний параметр	**Клас інтенсивності заявки**	**Значення**
λ	Низька	10 – 500 заявок в годину
	Середня	500 – 1500 заявок в годину
	Висока	Більше 1500 заявок в годину
$1/\delta_h$, $1/\delta_w$, $1/\delta_c$		1 – 5 секунд
Ph, Pw, Pc		0 – 1
N_{MN}		0 – 1000

Побудований модельний граф зображено на рис. 3.4. Для побудови використано функцію Matlab grPlot.

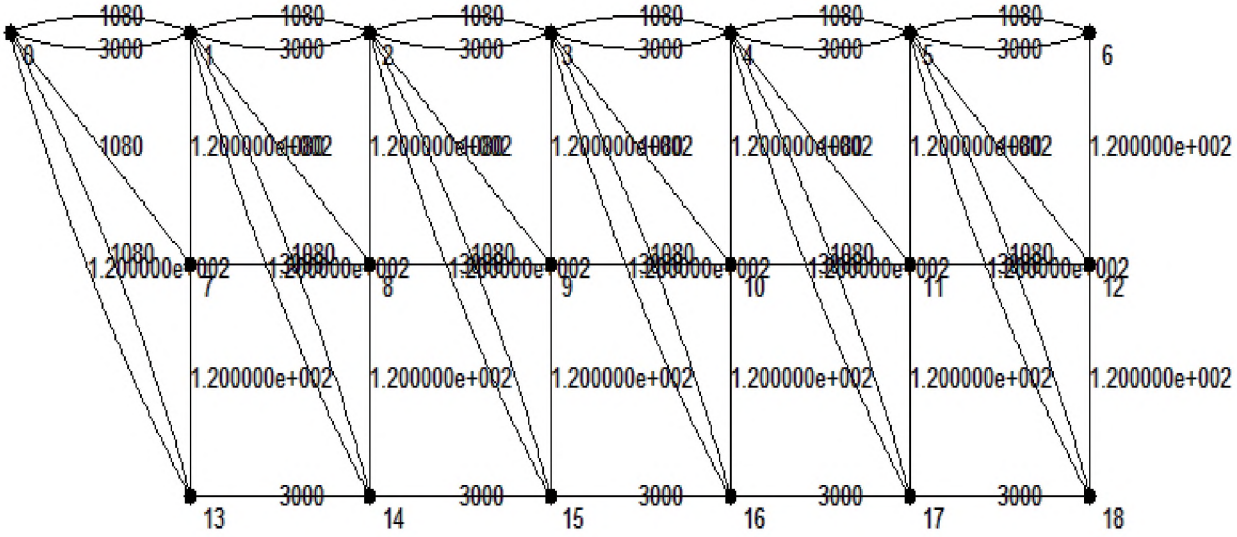

Рисунок 3.4 – Орієнтований граф моделі менеджера MN (N=5).

Після побудови розміченого графа моделі, згідно з принципом Колмогорова-Чепмена, потрібно скласти систему диференціальних рівнянь (СДР). Для різних завдань, таких як вивчення надійності та готовності системи або стійкості до зміни вхідних параметрів, потрібно розв'язувати СДР за допомогою чисельних методів. Інші завдання, такі як визначення показників якості обслуговування, вимагають визначення значень стаціонарної ймовірності Pi. Для цих завдань СДР перетворюються в систему лінійних рівнянь (СЛР) з урахуванням початкових умов об'єкта моделювання.

Перехід від СДР до СЛР виконується шляхом прирівнювання похідних dPi/dt до нуля, вважаючи Pi константами (які не залежать від часу t). Рішення отриманої СЛР здійснюється з використанням умови нормування, щоб уникнути безлічі рішень у системі.

Після отримання системи диференціальних рівнянь (СДР) та їхнього перетворення в систему лінійних рівнянь (СЛР) для забезпечення стаціонарності ймовірностей Pi, можна визначити ряд ключових параметрів моделі. Один з

важливих аспектів цього процесу – вибір методів розв'язання СЛР, оскільки це визначає точність і швидкість знаходження рішень. Наприклад, для визначення показників якості обслуговування може бути використано ітераційні методи, які потребують великої обчислювальної потужності.

Після визначення ймовірностей Pi для кожного стану можна визначити ряд показників моделі. До них відносяться:

Pblock – ймовірність відмови обслуговування через переповнення черги/буфера MN;

Pdrop – ймовірність відмови обслуговування через нестачу ресурсів в пулах фізичних серверів;

Preject – загальна ймовірність відмови обслуговування;

E[NMN] – середня кількість заявок у черзі менеджера;

E[Tq_dec] – середній час очікування прийнятої заявки у черзі менеджера;

E[Tdecision] – середній час пошуку вільного ресурсу у менеджері.

Формули для розрахунку цих показників наведені в таблиці 3.2.

Таблиця 3.2 – Вихідні показники моделі MN

Показник	**Прив'язка до вірогідності** Pi	**Розрахункова формула**
P_{block}	Крайні праві стани графа стану графа $P_{N,h}$ $P_{N,w}$ $P_{N,c}$	$\sum_{j\in h,w,c} P_{N,j}$
P_{drop}	Нижні стани графа $P_{0,c}$ $P_{1,c}$. $P_{N,c}$ з урахуванням інтенсивності відмови $(1-P_c)\delta_c$	$\frac{\delta_c(1-P_c)}{\lambda}\times\sum_{i=0}^{N} P_{i,c}$
P_{reject}	Pblock + Pdrop	
$E[N_{MN}]$	Сума творів вірогідності станів $P_{i,j}$ на i – кількість заявок в черзі, j={h, w, c}	$\sum_{i=1}^{N} i\cdot\left(P_{i,h}+P_{i,w}+P_{i,c}\right)$
$E[T_{q\_dec}]$	$E[N_{RPDE}] \Big/ \lambda\left(1-P_{reject}\right)$	
$E[T_{decision}]$	$\frac{E[N_{decision}]}{\lambda\left(1-P_{reject}\right)} = \frac{1-P_{0,0}}{\lambda\left(1-P_{reject}\right)}$	

Модель менеджера Grid системи реалізована у вигляді функціонального блоку середовища Matlab, код якого представлено у додатку А. Результати моделювання представлені у графічному вигляді на рис. 3.5 та рис. 3.6.

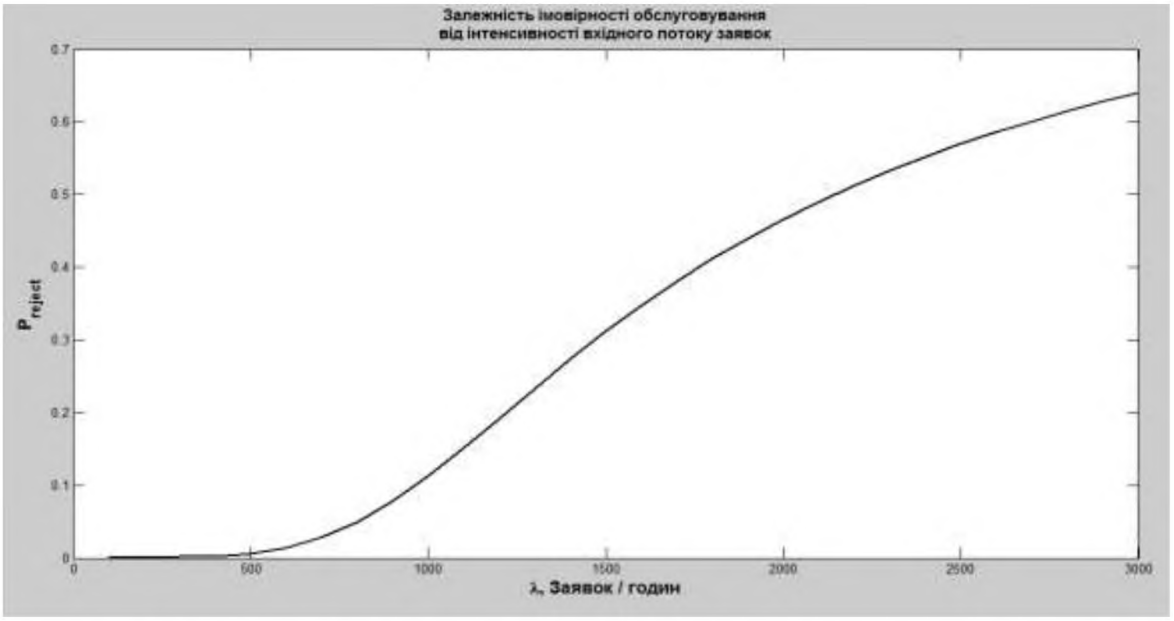

Рисунок 3.5 – Залежність ймовірності відмови в обслуговуванні від інтенсивності потоку вхідних заявок

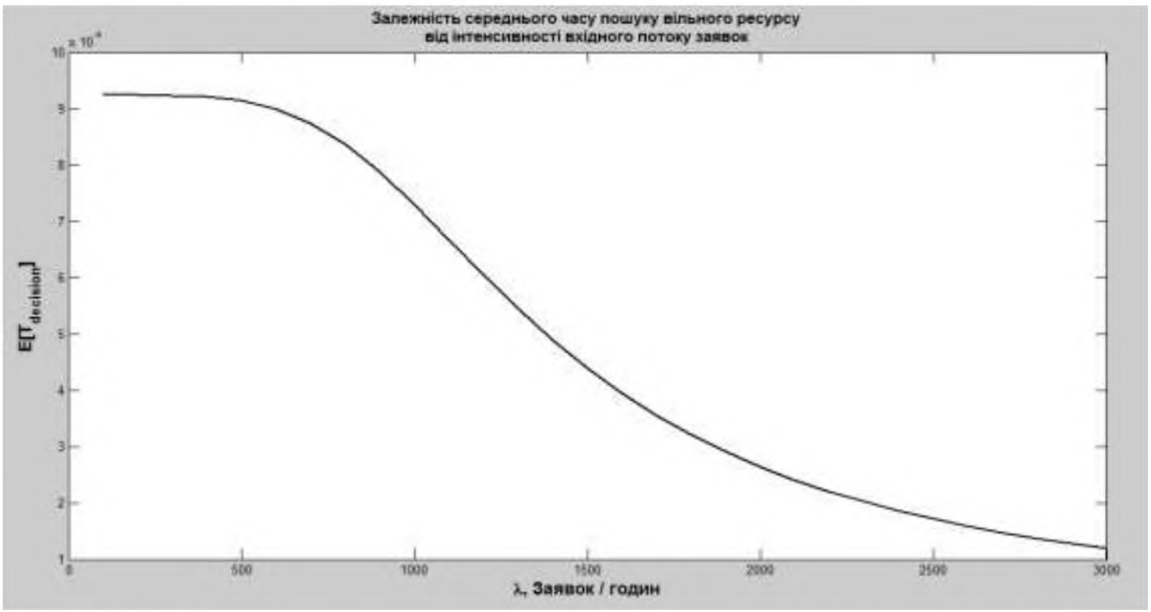

Рисунок 3.6 – Залежність затримки в обслуговуванні від інтенсивності потоку вхідних заявок

Після перегляду результатів обчислень що зображені на графіках рис. 3.5 та рис. 3.6 можна спостерігати, що після зростання кількості заявок більше 500 за годину починається «відкидання» заявок. Також спостерігається зменшення середнього часу пошуку вільного ресурсу від інтенсивності вхідного потоку заявок.

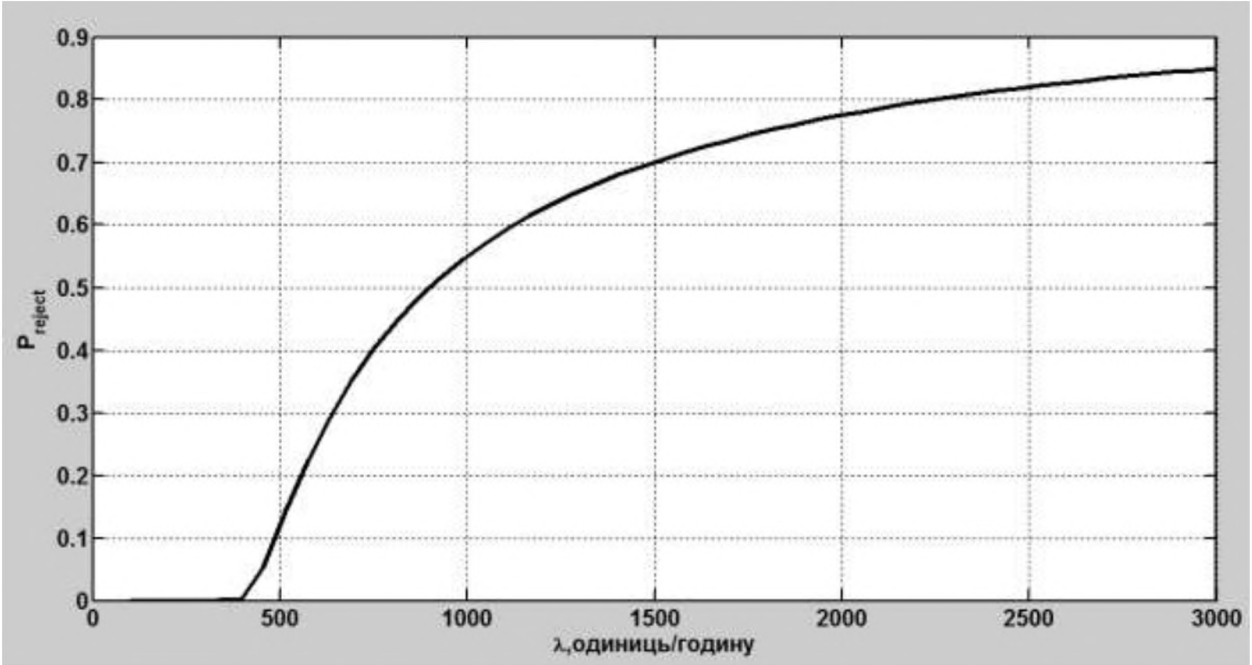

Рисунок 3.7 – Залежність імовірності відмови в обслуговувані від інтенсивності вхідного потоку заявок для моделі з трьох пулів серверів

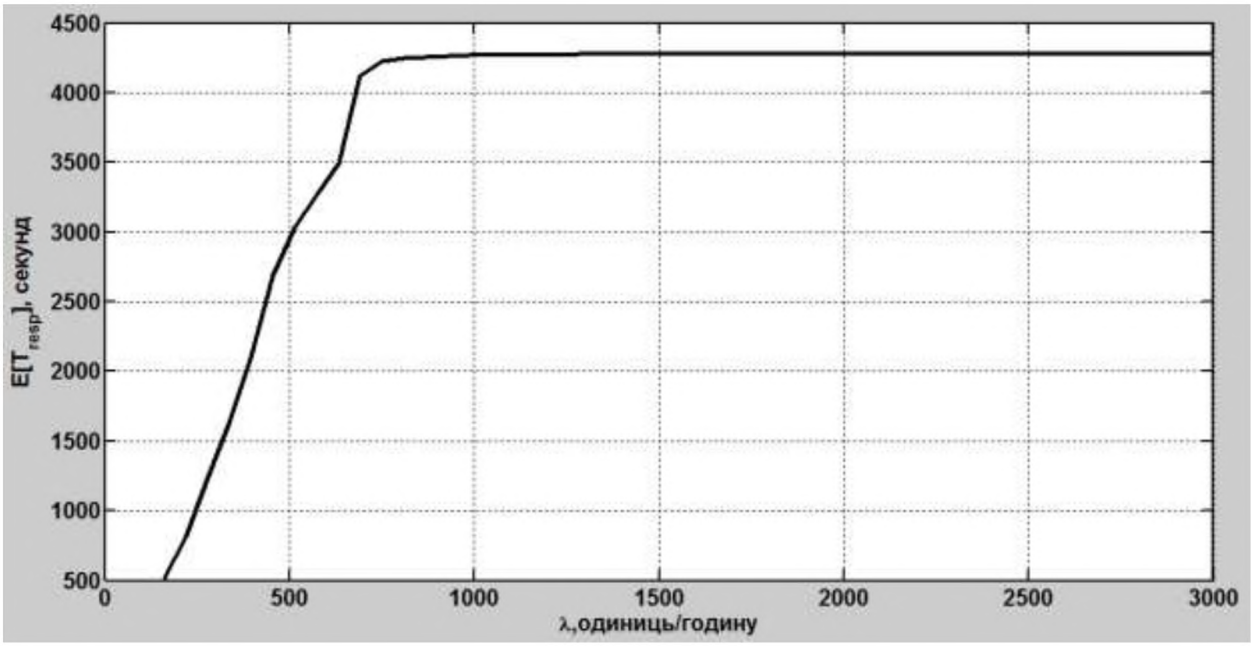

Рисунок 3.8 – Залежність середньої затримки до відгуку хмарного сервера від інтенсивності вхідного потоку заявок для моделі з трьох пулів серверів

Графіки моделі з трьох пулів серверів мають відмінність від попередньої моделі системи менеджера, оскільки ілюструють більш стрімке зростання

ймовірності відмови в обслуговуванні при збільшенні інтенсивності заявок понад 500 одиниць за годину, та вихід показника часової затримки в обслуговуванні на стаціонарний рівень в 4370 секунд.

3.4 Економічне обґрунтування використання Grid системи з кластеризованими ресурсами

Економічна ефективність визначається через відношення прибутковості виробництва до загальних витрат та використаних ресурсів. Якщо перший показник переважає другий, це означає досягнення поставлених цілей та задоволення потреб. Якщо ситуація навпаки, підприємство зазнає збитків, а економічного ефекту немає. Суть економічної ефективності полягає в максимізації результатів виробництва з обмежених ресурсів, витративши їхні покладені витрати.

Етап 1. Оцінка витрат на інформаційні технології. На цьому етапі визначається обсяг інвестицій в інформаційні технології для досягнення поставлених цілей.

Оцінка витрат на інформаційні технології включає два етапи:

1. Оцінювання витрат проєкту передбачає визначення всіх капітальних і поточних витрат, пов'язаних із впровадженням проєкту. Оцінка прямих витрат визначається за формулою:

$$B_П = B_{ТЗ} + B_{ППЗ} + B_{ОП} + B_{ВСЗ} + B_{ПСП} + B_У + B_{РПЗ} + B_І, \quad (3.1)$$

$$B_П = 16300+16000+10000+0+0+5300+0+1000=48600$$

де: витрати на технічне забезпечення ($В_{ТЗ}$) включають у себе придбання оперативної пам'яті, жорстких дисків, мережевих карт та термопасти;

– витрати на програмне забезпечення ($В_{ПЗ}$) включають у себе придбання серверної операційної системи Windows Server 2019;

– витрати на обслуговування проєкту ($В_{ОП}$) охоплюють витрати на створення, налаштування та адміністрування доменного сервера, здійснене системним адміністратором;

– витрати на відповідальність перед спільнотою ($В_{ВСЗ}$) включають соціальні заходи, але вони не є необхідними для створення доменного сервера;

– витрати на співпрацю з партнерами ($В_{ПСП}$) в даному випадку відсутні;

– витрати на інтернет ($В_{У}$) включають у себе підключення та проведення Інтернету за 500 грн. Потрібна також оплата тарифного плану 400 грн/міс, з річною платою в розмірі 5300 грн;

– витрати на програмне забезпечення ($В_{РПЗ}$) включають у себе витрати на всі програмні продукти, крім операційної системи, які є безкоштовними;

– витрати на навчання персоналу ($В_{І}$) включають у себе витрати на навчання персоналу з підключення до Grid серверу за 1000 грн.

2. Оцінка непрямих витрат на проєкт впровадження який визначається за формулою, грн:

$$B_H = B_{H1} + B_{H2}, \tag{3.2}$$

$$B_H = 1000+100=1100$$

де: B_{H1} – витрати пов’язані з простоями, тобто вихід з ладу доменного сервера 1000 грн за тестування;

B_{H2} – витрати пов’язані з людським фактором, наприклад пошкодження кабелю Ethernet, 100 грн за заміну.

3. Оцінка витрат на обслуговування доменного серверу за період його життєвого циклу.

$$B_{УТР} = B_{ОП} + B_{ВСЗ} + B_{П} + B_{І}, \tag{3.3}$$

$$B_{УТР} =10000+0+48600+2000=60600$$

де: $B_{ОП}$ – витрати на оплату зарплатні системному адміністратору;

$B_{ВСЗ}$ – соціальні заходи не використовувались;

$B_{П}$ – оцінка непрямих витрат 48,600; $B_{І}$ – витрати на вдосконалення сервера 2000 грн.

4. Визначення загальних витрат на проєкт буде розраховуватися за формулою:

$$B_{ІТ}=B_{П}+B_{Н}+B_{УТР}, \quad (3.4)$$

$$B_{ІТ} = 48600+1100+60600=110300.$$

Етап 2. Оцінка вимог використання інформаційних технологій.

Впровадження проєкту на підприємстві, виконується за допомого результатів операційної діяльності які здійснюється помісячно, приклад показано в табл. 3.3.

Таблиця 3.3 – Операційна діяльність по проєкту

Показники	Значення на кроці, тис. грн.	
	1	2
1. Ціна, грн./ш	10000	8000
2. Виручка, тис. грн.	50000	50000
4. Постійні витрати, тис. грн.	400	400
5. Амортизація устаткування, тис. грн.	800	800
7. Результат від операційної діяльності, тис. грн.	25296	25296

Дохід оподаткування розраховується за формулою:

$$\text{Пр} = \text{Виручка} – \text{Затрати} = 500000\text{-}110300 = 389700 \quad (3.5)$$

Податок на приріст:

$$\text{Под} = \text{Пр} \cdot \text{Ст}_{\text{под}} = \quad 389700*0,25 = 97425 \quad (3.6)$$

де: $Ст_{под}$ – ставка податку на приріст використовує стале значення 0,25

Чистий приріст складе:

$$Прч = Пр – Под = 389.700 – 97.425 = 292275 \quad (3.7)$$

Підсумок операційної діяльності буде складати:

$$CF_2(t) = Пр_ч + А = 292.275 + 1.600 = 293875 \quad (3.8)$$

Для розробки, впровадження та навчання персоналу доменного серверу потрібно затрати підприємству на даний момент 48600 грн.

Етап 3. На цьому етапі ми розрахуємо економічну ефективність проєкту по використанню Grid обчислень з кластеризованими ресурсами. Потрібно розрахувати чисту вартість при одноразовому здійснені інвестиційних витрат на початку здійснення проєкту. Розрахування буде здійснюватися за допомогою формули :

$$NPV = \sum_{t=1}^{n} \frac{CF_t}{(1+i)^t} – INV_0 \quad (3.9)$$

$$NPV = \sum_{t=1}^{n} \frac{250000_1}{(1+25)^1} – INV_0 = 450000$$

Наступним кроком буде показник бухгалтерської рентабельності інвестиційного проєкту (ROI):

$$ROI = \frac{AP}{(INV_1+INV_2)/2} * 100. \quad (3.10)$$

$$ROI = \frac{293,875}{(110,300_1+110,300_2)/2} * 100 = 266,432$$

Для розробки, впровадження та адміністрування серверу компанії доведеться витратити 110300 грн, що впорядкувати систему облікових записів та контролю за безпекою інформації.

Висновки до розділу 3

У третьому розділі була розглянута модель планування завдань у Grid системі з кластеризованими ресурсами. Виділено елементи Grid інфраструктури: менеджер, пули гарячих, теплих та холодних серверів та безпосередні виконавці – віртуальні машини. Як один з показників якості обслуговування розглянуто середній час до безпосереднього обслуговування заявки в хмарі. Розглянуто ролі учасників процесу надання послуг та особливості домовленостей між ними.

Розроблено та досліджено модель менеджера Grid системи як системи масового обслуговування з відмовами та обмеженою чергою. Підвищено точність математичної моделі шляхом розв'язку системи диференційних рівнянь Колмогорова-Чепмена за допомогою вирішувача ode15s до рівня 10^{-9}. Аналіз отриманих результатів моделювання Grid компоненти показав, після зростання кількості заявок понад 1000 за годину починається рівень ймовірності відмови збільшується до 0,1; а середній час пошуку вільного ресурсу зменшується.

Визначено порядок розрахунку економічної ефективності функціонування Grid системи з кластеризованими ресурсами. Якщо підприємство не бажає виділяти кошти для купівлі спеціального серверного обладнання для реалізації проєкту доменного сервера, сервер можна розгорнути на звичайному персональному комп'ютері, для цього потрібно встановити серверне програмне забезпечення, але непотрібно забувати, що доменний сервер повинен працювати цілодобово, не завжди звичайна комп'ютерна техніка може витримати такі навантаження.

Впровадження Grid обчислень є економічно вигідним, так як підприємство буде мати змогу скоротити витрати на програмне забезпечення, за допомогою якого відбувався захист інформації.

ВИСНОВКИ

У роботі була поставлена і вирішена актуальна задача розробки та дослідження моделі функціонування Grid системи з кластеризованими ресурсами для оптимізації планування завдань.

У ході виконання кваліфікаційної роботи було проведено дослідження стану розвитку галузі в сфері Grid обчислень та перспективи їх розвитку, розглянуто особливості рівнів Grid архітектури та протоколів. На основі роботи можна сформулювати наступні висновки:

1. Проведено огляд та аналіз наявних рішень для організації обчислювальних інфраструктур із кластеризованих комп'ютерів. Під час аналізу розглянуто чотири основних підходи: перший полягає у створенні проєктів, до яких підключаються комп'ютери – виконавці; другий використовує P2P-технології та об'єднання виконавчих комп'ютерів у однорангові мережі; третій характеризується системами із централізованим управлінням; а четвертий представлений приватними розробками корпоративних систем. Сформульовані вимоги до програмного забезпечення для однорівневих Grid систем.

2. Визначено архітектуру системи управління завданнями для Grid системи з кластеризованими ресурсами. Архітектура складається з трьох компонентів: диспетчера, агента та інтерфейсу користувача. Диспетчер реалізується через набір вебслужб та розташовується на виділеному комп'ютері у Grid сегменті. Ресурси виконавчих комп'ютерів Grid інфраструктури не доступні користувачам безпосередньо, оскільки всі зовнішні інтерфейси доступу до них централізовані в диспетчері.

3. Основна функція агента, що встановлюється на виконавчому комп'ютері, полягає у керуванні завданням на етапі виконання. Інтерфейс користувача дозволяє керувати власними завданнями у стандартному для Grid форматі та отримувати інформацію про їхній стан.

4. Розглянуто модель планування завдань у Grid системі з кластеризованими ресурсами. Виділено елементи Grid інфраструктури: менеджер, пули гарячих,

теплих та холодних серверів та безпосередні виконавці – віртуальні машини. Як один з показників якості обслуговування розглянуто середній час до безпосереднього обслуговування заявки в Grid. Розглянуто ролі учасників процесу надання послуг та особливості домовленостей між ними.

5. Розроблено та досліджено модель менеджера Grid системи як системи масового обслуговування з відмовами та обмеженою чергою. Підвищено точність математичної моделі шляхом розв'язку системи диференційних рівнянь Колмогорова-Чепмена за допомогою вирішувача ode15s до рівня 10^{-9}. Аналіз отриманих результатів моделювання Grid компоненти показав, після зростання кількості заявок понад 1000 за годину починається рівень ймовірності відмови збільшується до 0,1; а середній час пошуку вільного ресурсу зменшується.

6. Визначено порядок розрахунку економічної ефективності функціонування Grid системи з кластеризованими ресурсами. Якщо підприємство не бажає виділяти кошти для купівлі спеціального серверного обладнання для реалізації проєкту доменного сервера, сервер можна розгорнути на звичайному персональному комп'ютері, для цього потрібно встановити серверне програмне забезпечення, але непотрібно забувати, що доменний сервер повинен працювати цілодобово, не завжди звичайна комп'ютерна техніка може витримати такі навантаження.

Таким чином, поставлені задачі розв'язано у повному обсязі. Напрямком подальших досліджень є розробка та дослідження практичних рекомендацій щодо розгортання Grid системи з кластеризованими ресурсами в навчальній лабораторії вищого навчального закладу.