Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Післядипломної освіти_____
(повна назва)

Кафедра _____Програмної інженерії_____
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

_____другий (магістерський)_____
(рівень вищої освіти)

Дослідження методів оптимізації обчислень у хмарних технологіях
(тема)

Виконав: студент 2 курсу, групи ІПЗмзд-18-1
спеціальності 121- Інженерія програмного забезпечення_____
(код і повна назва спеціальності)
освітньо-професійної програми Інженерія програмного забезпечення _____
(повна назва освітньої програми)

_____ Кулик В.В._____
(прізвище, ініціали)
Керівник _____к.т.н., доц. Назаров О.С.___
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

2020 р.

Харківський національний університет радіоелектроніки

Факультет_____Післядипломної освіти_____
Кафедра_____Програмної інженерії_____
Рівень вищої освіти_____другий (магістерський)_____

Спеціальність _____121- Інженерія програмного забезпечення_____
(код і повна назва)
Освітньо-професійна програма___Інженерія програмного забезпечення____
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри_____
(підпис)
«_____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ
студентові _____Кулику Владиславу Володимировичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи _Дослідження методів оптимізації обчислень у хмарних технологіях_____
затверджена наказом по університету від "___" _____20 ___ р № _____
2. Термін подання студентом роботи до екзаменаційної комісії
_____ 2020р.
3. Вихідні дані до роботи *методи обчислень у хмарних технологіях, пояснювальна записка, порівняльна характеристика для наведених хмарних сервісів.*
4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз проблемної галузі і постановка задачі, огляд методів обчислень у хмарних технологіях, порівняння існуючих хмарних сервісів.*
_____
5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) *рисунки, таблиці, слайди презентації*_____

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	к.т.н., доц. Назаров О.С.		

КАЛЕНДАРНИЙ ПЛАН

	Назва етапів дипломного проекту	Термін виконання етапів проекту (тиждень)	Примітка
1	Аналіз предметної галузі	15.03.2020	виконано
2	Огляд існуючих методів	01.04.2020	виконано
4	Розробка порівняльної характеристики	20.04.2020	виконано
5	Підготовка пояснювальної записки	24.04.2020	виконано
6	Спецчастина	30.04.2020	виконано
7	Підготовка презентації та доповіді	02.05.2020	виконано
8	Нормоконтроль, рецензування	20.05.2020	виконано
9	Попередній захист	25.05.2020	виконано
10	Занесення диплома в електронний архів	25.05.2020	виконано

Дата видачі завдання _____

Студент гр. ІПЗмзд-18-1 _____ Кулик В.В.

Керівник _____ к.т.н., доц. Назаров О.С.

РЕФЕРАТ / ABSTRACT

Атестаційна робота магістра містить: 71 с., 20 рис., 3 табл., 2 додатки, 22 джерела.

ХМАРНІ ОБЧИСЛЕННЯ, ВІРТУАЛЬНІ МАШИНИ, ПРИВАТНІ ХМАРИ, AMAZON ВЕБ СЕРВІСИ, AZURE ХМАРА, ГІБРИДНА ХМАРА, ПУБЛІЧНА ХМАРА, ВІРТУАЛІЗАЦІЯ.

Метою роботи є дослідження методів оптимізації обчислень у хмарних технологіях.

В ході роботи було проаналізовано методи естимації витрат на моніторинг хмарної архітектури, методи оптимізації хмарної інфраструктури, проведено порівняльну характеристику віртуального серверу проти хмарного сервісу, а також різні методи автоматизації процесів.

CLOUD COMPUTING, VIRTUAL MACHINES, PRIVATE CLOUDS, AMAZON WEB SERVICES, AZURE CLOUD, HYBRID CLOUD, PUBLIC CLOUD, VIRTUALIZATION.

The object of research is to study the methods of optimization of calculations in cloud technologies.

In the course of work the methods of estimation of expenses for monitoring of cloud architecture, methods of optimization of cloud infrastructure, the comparative characteristic of the virtual server of simple cloud service, and also various methods of automation of processes were analyzed.

ЗМІСТ

ВСТУП

Останніми роками рівень комп'ютеризації підвищується у всіх сферах життєдіяльності людини. Це трапляється на підприємствах, у державних установах і навіть удома. Сьогодні людина не повинна навіть виходити з дому, щоб отримати бажане, оскільки Інтернет надає можливість отримання доступу, якщо не до всієї, то хоча б до більшості необхідної інформації та послуг.

Розробка будь-якого програмного забезпечення на даний час проходить у умовах чітких обмежень за рахунок бюджету компанії-замовника, що сильно обмежує можливості розгортання інфраструктури для вдалої роботи додатку.

Коли замовник замовляє розробку продукту, компанія-виконавець повинна чітко зрозуміти обсяг навантаження системи та фінансові можливості клієнта, оскільки таким чином можна зрозуміти у яких межах можливо проводити розробку продукту і чи взагалі можливо розробити додаток, який буде відповідати умовам доступності, відмовостійкості та ефективності використання ресурсів.

Варто відзначити, що за останні роки набувають популярності хмарні технології та хмарні обчислення. Все частіше програмісти використовують сервіси провайдерів хмарних технологій, таких як Amazon Web Services, Azure, Google Cloud Platform і так далі. Так як це комерційні організації, то будь-який бізнес, який використовує «хмари» и своєму програмному рішенні платить за це гроші. Тому особливим стає питання як краще організовувати архітектуру проекту з точку зору фінансів: самому розробляти архітектуру середовища розробки, чи звертатись до провайдерів хмарних технологій.

У сучасному світі корпоративних IT є багато факторів, які компанія повинна врахувати, щоб вирішити, чи потрібна хмарна інфраструктура. І навпаки, існує багато компаній, які не в змозі здійснити стрибок у хмару,

замість того, вони готові покладатися на свої перевірені спадкові «On-Premise» системи для ведення бізнесу.

Людство вже більше десяти років переживає епоху хмарних обчислень, і, нажаль, на даний момент не вистачає думок, які би проаналізували чому слід переміщувати навантаження на хмару. Що рідше - це розмірене обговорення того, чому в деяких випадках ви можете не переносити робоче навантаження в хмару - або "репатріювати" те, яке ви вже перемістили туди. У результаті ця робота має допомогти віднайти чіткі критерії для перенесення своїх обчислень у хмару.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Історія технології хмарних обчислень

Для того, щоб краще зрозуміти проблеми, які виникли під час розвитку технологій обчислень, необхідно оглянути історію становлення «хмари» як нового виду обчислень. Хмарні сервіси, що дозволяють перенести обчислювальні ресурси та дані на віддалені сервери, в останні роки стали одним з основних трендів розвитку IT-технологій.

Концепція хмарних обчислень з'явилася ще в 1960 році, коли американський учений, фахівець з теорії ЕОМ Джон Маккарті (John McCarthy) висловив припущення, що коли-небудь комп'ютерні обчислення стануть надаватися подібно комунальним послугам (public utility). Розповсюдження мереж з високою потужністю, низька вартість комп'ютерів і пристроїв зберігання даних, а також широке впровадження віртуалізації, сервіс-орієнтованої архітектури привели до величезного зростання хмарних обчислень. Кінцеві користувачі можуть не перейматися роботою обладнання технологічної інфраструктури «в хмарі», яка їх підтримує. Аналогією обчислювальних «хмар» зі звичного життя можуть служити електростанції. Хоча домовласник може купити електрогенератор і піклуватися про його справність самостійно, більшість людей вважає за краще отримувати енергію від централізованих постачальників[1].

У 1969 р. Дж. К. Р. Ліклайдер допоміг розробити прототип Інтернету – ARPANET (мережа розширених дослідницьких проектів). JCR був і психологом, і комп'ютерним науковцем, і пропагував бачення під назвою «Міжгалактична комп’ютерна мережа», в якому всі на планеті будуть пов'язані між собою за допомогою комп’ютерів і матимуть доступ до інформації з будь-якого місця. Міжгалактична комп'ютерна мережа, інакше відома як Інтернет, необхідна для доступу до Хмари [2].

Сенс віртуалізації почав змінюватися в 1970-х, і тепер описується створення віртуальної машини, яка діє як справжній комп'ютер, з повністю функціональною операційною системою. Концепція віртуалізації еволюціонувала з Інтернетом, коли підприємства почали пропонувати «віртуальні» приватні мережі як послугу оренди. Використання віртуальних комп'ютерів стало популярним у 90-х роках, що призвело до розвитку сучасної інфраструктури хмарних обчислень.

На своїх ранніх етапах Хмара використовувалася для вираження порожнього простору між кінцевим користувачем та провайдером. У 1997 році професор Рамнат Челлапа з Університету Еморі визначив Хмарні обчислення як нову "обчислювальну парадигму, де межі обчислень визначатимуться економічним обґрунтуванням, а не лише технічними обмеженнями". Цей дещо складний опис відповідає дійсності опису еволюції Хмари.

Хмара набула популярності, коли компанії краще розуміли її послуги та корисність. У 1999 році SalesForce став популярним прикладом успішного використання хмарних обчислень. Вони використовували його для того, щоб почати розробляти ідею використання Інтернету для доставки програмних програм кінцевим споживачам. До програми (або програми) може бути доступний і завантажений будь-хто, хто має доступ до Інтернету. Підприємства можуть придбати програмне забезпечення на вимогу, економічно вигідно, не виходячи з офісу [3].

Сам термін «хмара» походить з телефонії, тому що телекомунікаційні компанії, які до 1990-х років пропонували в основному виділені схеми передачі «точка-точка», почали пропонувати віртуальні приватні мережі (VPN), з порівняною якістю обслуговування, але при набагато менших витратах. Перемикаючи трафік для оптимального використання каналів вони мали змогу ефективніше використовувати мережу. Символ хмари був використаний для позначення розмежування між користувачем і постачальником [4].

Ключову роль в розвитку хмарних обчислень зіграв Amazon. У 2002 році Amazon представила свої веб-роздрібні послуги. Це був перший великий бізнес, який вирішив використати лише 10% своєї потужності (що було звичним для того часу) як проблему, яку потрібно вирішити. Модель інфраструктури хмарних обчислень надала їм можливість більш ефективно використовувати потужність свого комп'ютера. Незабаром інші великі організації наслідували їх приклад. У 2006 році Amazon запустив веб-сервіси Amazon, які пропонують онлайн-сервіси для інших веб-сайтів або клієнтів. Один із веб-сайтів Amazon Web Services, який називається Amazon Mechanical Turk, пропонує різноманітні хмарні сервіси, включаючи зберігання, обчислення та "інтелект людини". Ще один із веб-сайтів Amazon Web Services - Elastic Compute Cloud (EC2), що дозволяє людям орендувати віртуальні комп'ютери та користуватися власними програмами та програмами.

У 2007 році IBM, Google та кілька університетів об'єднали зусилля для розробки ферми серверів для науково-дослідних проектів, що потребують як швидких процесорів, так і величезних наборів даних. Університет Вашингтона першим зареєстрував та використав ресурси, надані IBM та Google. Університет Карнегі Меллон, MIT, Стенфордський університет, Меріленський університет і Каліфорнійський університет в Берклі, швидко пішли за цим прикладом. Університети негайно зрозуміли, що комп'ютерні експерименти можна зробити швидше і за менші гроші, якщо IBM і Google підтримували їх дослідження. Оскільки велика частина досліджень була зосереджена на проблемах, які IBM та Google мали інтереси, вони також отримали користь від цієї домовленості. 2007 рік також був роком, коли Netflix запустив послугу потокового відео, використовуючи хмару, та надав підтримку практиці "binge-watching".

У 2008 році Eucalyptus запропонував першу сумісну платформу AWS API, яка використовувалася для розповсюдження приватних хмар. У тому ж році OpenNebula NASA представила перше програмне забезпечення з відкритим кодом для розгортання приватних та гібридних хмар. Багато його

найбільш інноваційних функцій були зосереджені на потребах великого бізнесу. Тоді принцип роботи хмарних технологій вже був сформований та зрозумілий для спеціалістів області інформаційних технологій.

У 2011 році IBM представила рамку IBM SmartCloud на підтримку Smarter Planet (проект культурного мислення). Тоді Apple запустила ICloud, який зосереджується на збереженні більшої кількості особистої інформації (фотографій, музики, відео тощо). Крім того, протягом цього року Microsoft почала рекламувати Хмару на телебаченні, даючи широкій публіці свою здатність зберігати фотографії чи відео з легким доступом.

Oracle представила Oracle Cloud у 2012 році, пропонуючи три основи для бізнесу: IaaS (інфраструктура як послуга), PaaS (платформа як послуга) та SAAS (програмне забезпечення як послуга).На початку 2008 року Eucalyptus став першою API-сумісною платформою з відкритим кодом для розгортання приватної хмари. На початку 2008 року OpenNebula став першим проектом з відкритим кодом для розгортання приватних і гібридних хмар.

1.2 Огляд сучасних тенденцій технології

Провідні постачальники хмарних технологій на 2020 рік зберегли свої позиції, але теми, стратегії та підходи до ринку змінюються. Боротьба за інфраструктуру як послугу великою мірою вирішені: першість перейшла до веб-служб Amazon, Microsoft Azure та Google Cloud Platform, але нові технології, такі як штучний інтелект та машинне навчання, віднайшли нову сферу для нових провайдерів.

Тим часом ринок хмарних обчислень у 2020 році вирішально мультихмарний спінінг, оскільки гібридна зміна таких гігантів, як IBM, яка придбала Red Hat, може змінити ринок. Цьогорічне видання провідних постачальників хмарних обчислень також містить гігантів «програмного

забезпечення як послуги», які все більше і більше керуватимуть операціями через розширення.

Перед оглядом типів інфраструктур та аналізу проблеми хмарних обчислень варто оглянути принцип роботи хмарних обчислень. Наглядно принцип зображений на рисунку 1.1 у вигляді схеми.

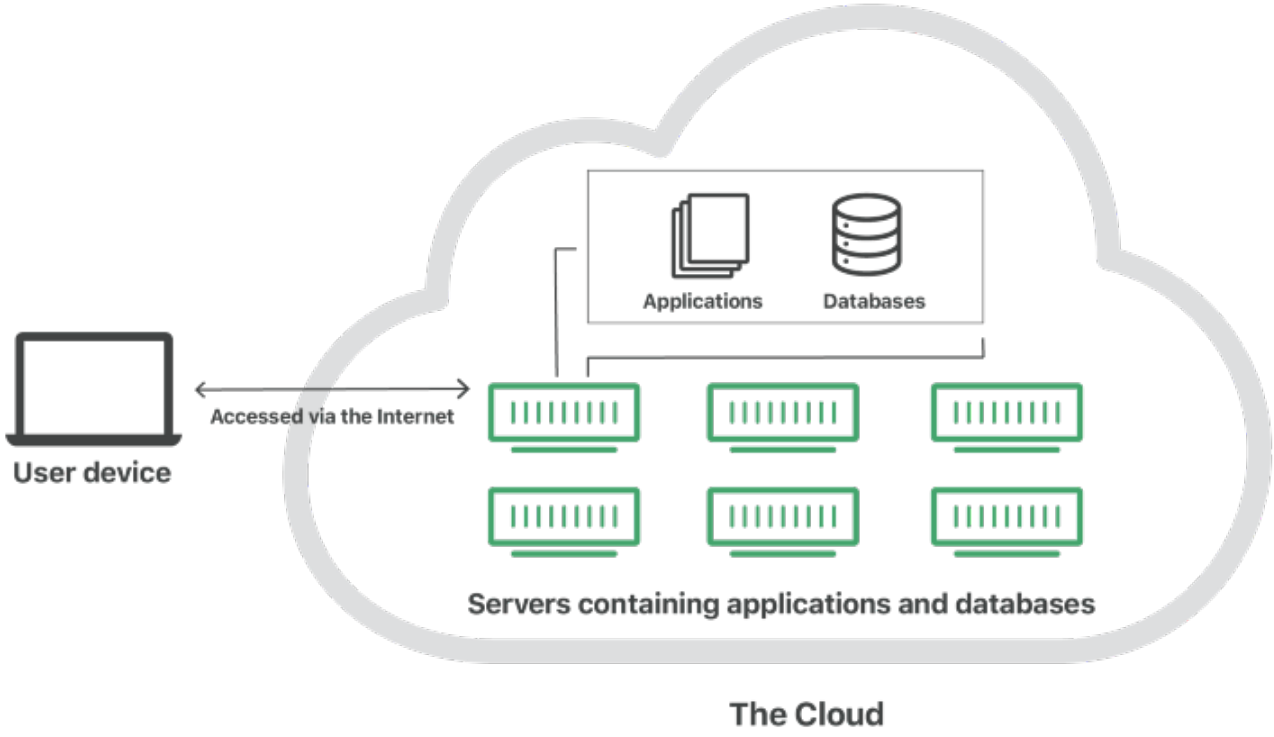

Рисунок 1.1 – Принцип роботи хмарних обчислень

Варто зауважити щодо хмари в 2020 році, що ринок не є нульовою сумою. Хмарні обчислення призводять до загальних витрат у сфері інформаційних технологій. Gartner прогнозує, що світові витрати на інформаційні технології в 2020 році отримають збільшення на 3,2% до 3,76 трлн доларів[5].

Насправді цілком можливо, що велике підприємство буде споживати послуги хмарних обчислень у кожного постачальника цього посібника. Справжня інновація у хмарі може бути від клієнтів, які унікальними способами поєднують та узгоджують наступних публічних постачальників хмар.

Виділені основні характеристики 2020 року, які слід розглянути серед провідних постачальників хмарних технологій:

– цінова потужність;
– комбінація моделей ліцензій;
– мультихмара;
– штучний інтелект та аналітика;
– IoT.

Цінова потужність характеризується на прикладі компанії Google. Було підвищено ціни на G-Suite та хмарний простір – технологію, де розташовані застосування для більшості нових технологій. Хоча послуги з обчислення та зберігання часто є гонкою донизу, інструменти для машинного навчання, штучного інтелекту та безсерверних функцій можуть скластися. Є вагома причина, що управління витратами - така велика тема для клієнтів хмарних обчислень - це, мабуть, найбільша проблема. Шукайте управління витратами та занепокоєння щодо блокування, щоб стати великими темами. Дійсно, опитування RightScale показало, що оптимізація витрат у хмарі є великим пріоритетом для великих компаній аж до малого бізнесу. Прийняття контейнерів може також сприяти питанням оптимізації витрат.

Хмарні обчислення використовувались з рівною вартістю на основі передплати та використання, але Microsoft рухалася до змішування моделей. Імовірно, що інші передові постальчиники також планують відповідну концепцію розповсюдження.

Дані Kentik стверджують, що користувачі хмарних технологій все частіше використовують послуги більше ніж одного постачальника. AWS та Microsoft Azure найчастіше спарюються. Google Cloud Platform також знаходиться в поєднанні. І, звичайно, ці постачальники публічних хмарних послуг часто пов'язані з існуючим центром обробки даних та приватними хмарними ресурсами. Додайте це, і там ведеться здорова гібридна та приватна хмарна гонка, і це впорядкувало заяву про підрядність. Мультихмарний підхід забезпечується віртуальними машинами та контейнерами [6].

Штучний інтелект, Інтернет речей та аналітика - це найвищі технології від постачальників хмарних технологій. Microsoft Azure, веб-сервіси Amazon та платформа Google мають подібні стратегії для виходу клієнтів на обчислення, хмарне сховище, безсерверні функції, а потім перепродають клієнтам інтелектуальний інтерфейс, який їх диференціює.

«Хмарно-обчислювальний ландшафт» швидко дозріває, але фінансова прозорість сумнівна. Вона є нижчою серед постачальників хмарних технологій. Наприклад, Oracle виводив інфраструктури, платформи та програмне забезпечення як послугу до своїх фінансових звітів. Сьогодні хмарний бізнес Oracle об'єднаний разом. Microsoft має "комерційну хмару", яка дуже успішна, але також складна для розбору. IBM має дохід від хмарних ситуацій та дохід "під час обслуговування". Google взагалі не розподіляє дохід від наданих послуг. Проте аналізувати хмарні продажі стало складніше[7].

Тому можна умовно поділити гігантів індустрії на чотирьох великих постачальників інфраструктури, гібридних гравців та натовпу SaaS. Ця категоризація підштовхнула IBM до розвитку до рівня гіганта надання інфраструктури як послуги, до твінера, який охоплює інфраструктуру, платформу та програмне забезпечення. IBM є найбільш приватною хмарою та гібридом з гачками в IBM Cloud, а також іншими хмарними середовищами.

1.3 Основні принципи надання хмарних послуг

Концепція хмарних сервісів полягає в тому, що деяка кількість складових елементів залишається під власним управлінням і, відповідно, деяка – передається на аутсорінг та обслуговування сторонньою організацією.

IT-послуги можуть надаватися за схемою, коли клієнт не несе капітальних витрат, а платить тільки коли використовує сервіс. Компанія не організовує власний дата-центр, а орендує обчислювальні потужності у

хмарного провайдера. В такому випадку програмне забезпечення або інфраструктура надаються як сервіс. Для позначення такої моделі послуг прийнято визначення XaaS або "що завгодно як сервіс" (Anything-as-a-service). Під нього підпадають всі послуги, які виявляються через мережу Інтернет із застосуванням хмарних обчислень. "X" у цій абревіатурі означає невідому змінну, подібно до математичного рівняння. Якщо у якості сервіса надається інфраструктура, то вид послуг називається IaaS, якщо платформа для розробки – PaaS, якщо програмне забезпечення – SaaS [8].

Основними типами надання послуг є наступні:

– IaaS;

– PaaS;

– SaaS.

Між вказаними типами є наступна відмінність: в IaaS клієнт отримує тільки інфраструктуру, в PaaS – інфраструктуру і готове для розробки програмне забезпечення, в SaaS – готове працююче рішення в хмарі. Схематичне зображення типів постачання зображене на рисунку 1.2.

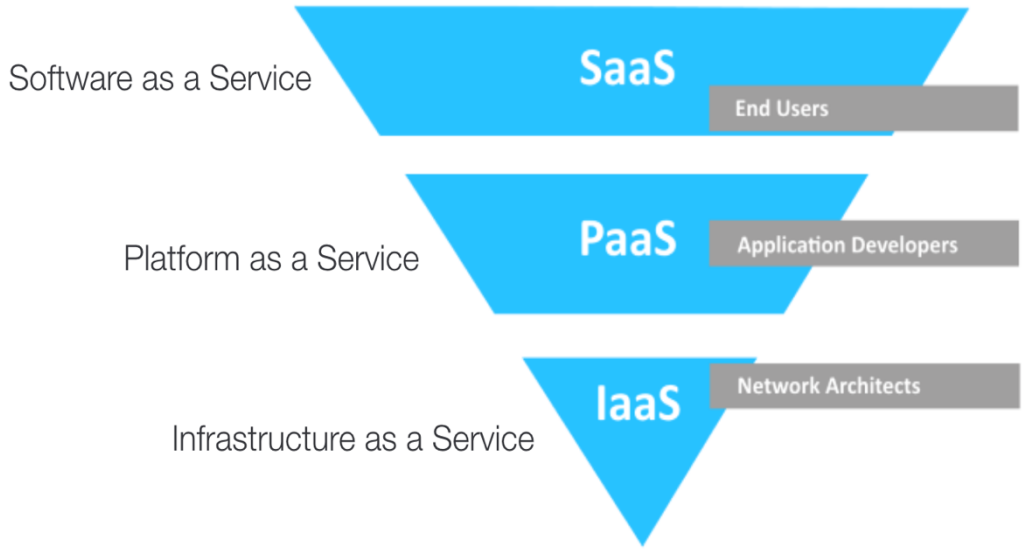

Рисунок 1.2 – Піраміда типів постачання послуг

Розглянемо наведені моделі постачання послуг більше детально.

Інфраструктура як послуга (англ. Infrastructure-as-a-Service; IaaS) – модель, за якою споживачі забезпечуються фундаментальними інформаційно-технологічними ресурсами – віртуальними серверами із заданою обчислювальною потужністю, операційною системою (визначеною провайдером) та доступом до мережі [9].

При використанні моделі «інфраструктура як послуга» клієнт, як правило, отримує серверний час, помножений на кількість задіяних віртуальних процесорів і віртуальних обсягів пам'яті, а також простір зберігання (з тарифікацією залежно від продуктивності), задану мережеву пропускну здатність, іноді – мережевий трафік.

IaaS знаходиться на найнижчому рівні серед моделей. На відміну від моделі PaaS, що постачає готове програмне забезпечення, його СУБД та засоби розробки та супроводу, та SaaS, на якому надається прикладне програмне забезпечення, в IaaS не передбачений контроль з боку постачальника за встановленими програмним забезпеченням, проте виконується контроль фізичної та віртуальної інфраструктури.

Програмне забезпечення як послуга (SaaS) надає клієнтові можливість використання систему постачальника, що функціонує на інфраструктурі. Доступ до програмного забезпечення здійснюється з різних клієнтських пристроїв через інтерфейс тонкого клієнта, наприклад, веб-браузер. Споживач не контролює і не керує хмарною інфраструктурою, на якій виконується програмне забезпечення, включаючи мережу, сервери, операційні системи, сховища даних і параметри програми.

Платформа як послуга (PaaS) – модель надання технології хмарних обчислень, за якою споживач отримує доступ інформаційно-технологічних платформ: операційних систем, систем управління базами даних, програмного забезпечення, засобів розробки і тестування, що розміщені на стороні провайдера. За цією моделлю інформаційно-технологічна інфраструктура, включаючи обчислювальні мережі та сервери, повністю керується провайдером. Також провайдер визначає доступні споживачеві види платформ

і набір їх керованих параметрів. При цьому споживач має можливість використовувати платформи, створювати їх віртуальні екземпляри, розробляти, тестувати та експлуатувати на них прикладне програмне забезпечення, при цьому динамічно модифікуючи кількість спожитих обчислювальних ресурсів.

Провайдер хмарної платформи стягує плату залежно від рівня споживання, тарифікація можлива за часом роботи додатків споживача, за обсягом оброблених даних і кількості транзакцій над ними. Порівняння моделей за набором послуг, що оплачуються, наведено на рисунку 1.3.

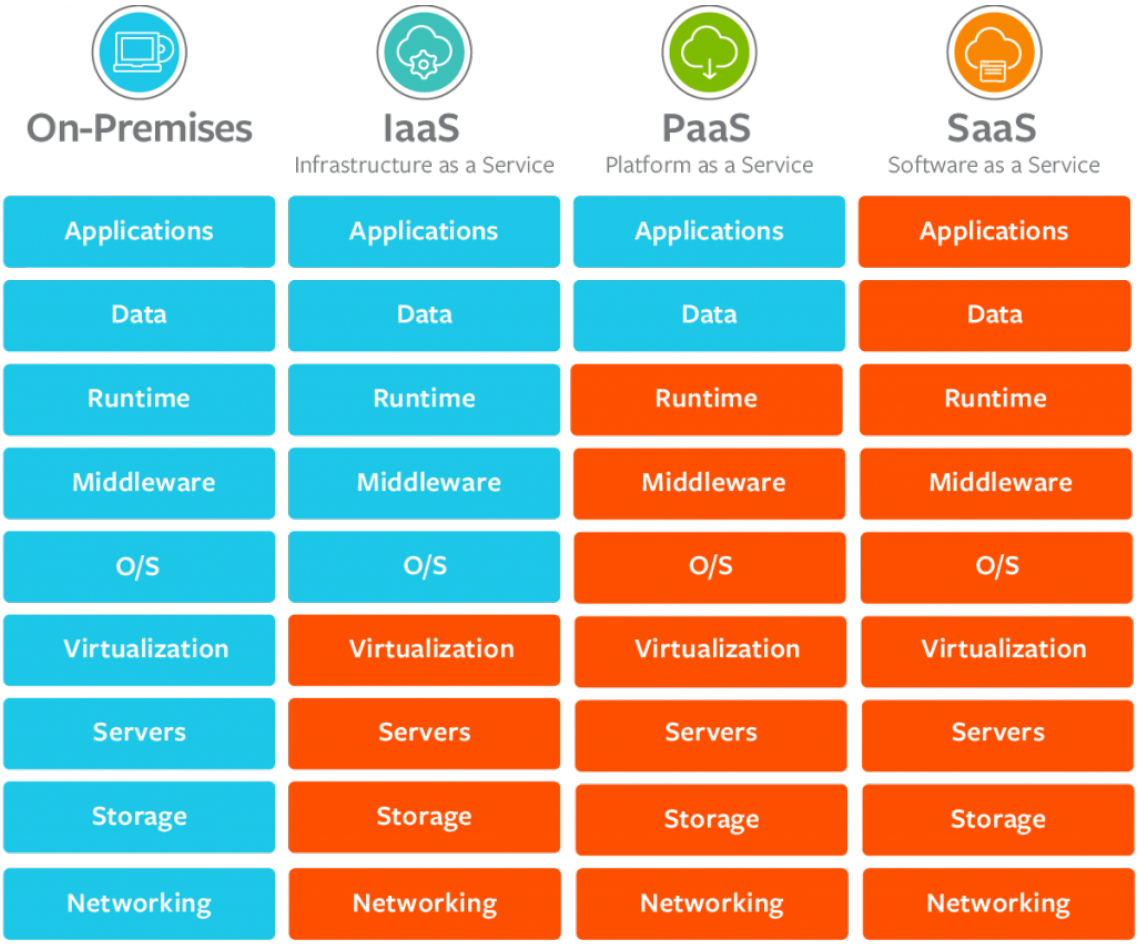

Рисунок 1.3 – Різниця в принципах надання послуг за різними моделями

Провайдери хмарних платформ досягають економічного ефекту за рахунок використання віртуалізації і економії на масштабах, коли з безлічі

споживачів в один і той же час лише частина з них активно використовує обчислювальні ресурси.

Також хмари можна поділити на три умовні моделі розгортання: Pulic Cloud, Private Cloud та Hybrid Cloud.

Приватна хмара - це хмарна інфраструктура, що призначена для використання виключно однією організацією з декількома користувачами (осіб, підрозділів, робочих місць). Приватна хмара може перебувати у власності, керуванні та експлуатації як самої організації, так і третьої сторони (чи їх комбінації). Така хмара може фізично знаходитись як в, так і поза юрисдикцією власника.

Публічна хмара - це інфраструктура для вільного використання. Публічна хмара може перебувати у власності, керуванні та експлуатації комерційних, академічних або державних установ. Публічна хмара перебуває в юрисдикції постачальника хмарних послуг.

Гібридна хмара - це інфраструктура, що складається з двох або більше різних хмарних інфраструктур, які залишаються унікальними структурними одиницями, але поєднуються між собою стандартизованими або приватними технологіями, що забезпечує переносимість даних та прикладних програм.

2 ПОСТАНОВКА ЗАДАЧІ

2.1 Проблема оптимізації хмарних обчислень з точки зору учасників бізнес-процесу

Модель хмарних обчислень дозволила користувачам змішувати власні витрати в операційній сфері, передаючи функції побудови, розробки та експлуатації центрів обробки даних (DPC) спеціалістам. Цей підхід особливо актуальний для компаній середнього та малого бізнесу (SMB), яким може не вистачати компетентного персоналу та ресурсів у цій галузі для налаштування та підтримки апаратного та програмного забезпечення. Створення цієї моделі стало можливим завдяки наступним двом технологіям:

– віртуалізація - процес представлення фізичних ресурсів у вигляді набору логічних ресурсів, в якому зберігаються всі властивості вихідних ресурсів [10]. Такий підхід дозволяє відійти від обмежень, пов'язаних із впровадженням, географічним розташуванням, конфігурацією та організувати ізольованість обчислювальних процесів та ресурсів;

– надання програмного забезпечення як послуги (SaaS) - бізнес-модель продажу програмного забезпечення, в якій виконання, розгортання та адміністрування здійснюються віддалено через Інтернет.

Сьогодні основна мотивація переходу компаній на хмарні послуги – економічна. Послуги, що надаються бізнес-моделлю SaaS, можуть бути доступні навіть компаніям, які не мають ресурсів для налаштування та обслуговування програмного забезпечення, а також працівникам, що мають компетенції в цій галузі діяльності. Перш за все, це компанії SMB, для яких використання хмарних обчислень дозволяє уникнути додаткових витрат, пов’язаних із встановленням та обслуговуванням обладнання та технічною підтримкою програмного забезпечення. Усі ці функції приймає постачальник послуги.

Виключення капітальних витрат та низьких експлуатаційних витрат при використанні хмарних обчислень надають компаніям можливість спробувати перші послуги перед тим, як впровадити їх у власні бізнес-процеси. У той же час прозорі моделі ціноутворення та гнучкі засоби збільшення / зменшення ресурсів обслуговування дозволяють заздалегідь планувати можливі витрати компанії. Цей підхід особливо актуальний для молодих компаній, які тільки починають розвивати свій бізнес.

Згідно з дослідженнями компанії Parallels, результат яких відповідає рисунку 2.1, попит на послуги SaaS зростатиме, і до 2015 року частка послуг SaaS серед усіх IT-рішень компаній SMB становитиме 67%.

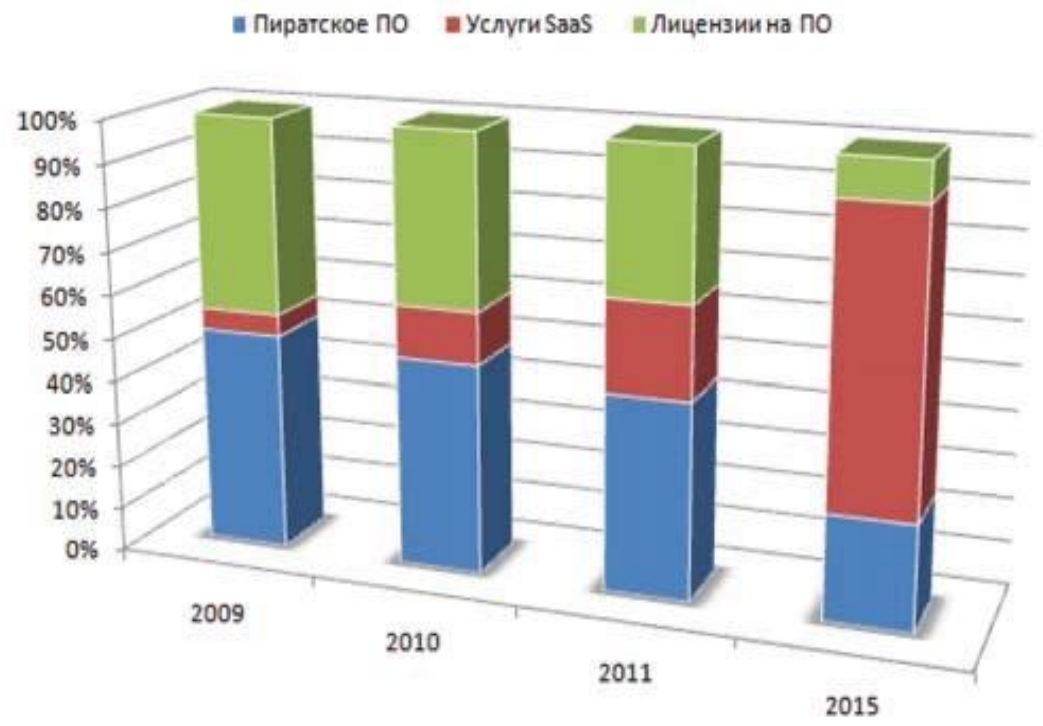

Рисунок 2.1 – Частка витрат малих підприємств на IT-продукцію

В даний час хмарні технології активно розвиваються по всьому світу. З точки зору Національного інституту стандартів та технологій США (NIST, Національний інститут стандартів і технологій) сучасні хмарні обчислення характеризуються такими властивостями:

– самообслуговування за запитом (On-Demand Self-Service) - надання користувачеві можливості самостійно визначати та змінювати потреби в ресурсах (продуктивність процесора, швидкість доступу, обсяг даних тощо);

– універсальний доступ (широкий доступ до мережі) - надання користувачам доступу через мережу передачі даних, незалежно від використовуваних термінальних пристроїв;

– єдиний пул ресурсів (Resource Pooling) - об'єднання ресурсів для обслуговування великої кількості користувачів з можливістю динамічного перерозподілу потужностей;

– швидка еластичність - надає користувачам можливість в будь-який час звузити або розширити послуги за рахунок одного або декількох ресурсів без додаткових витрат на взаємодію з постачальником;

– вимірювана послуга - автоматичне вимірювання споживаних ресурсів (наприклад, кількість збережених даних, пропускна здатність, кількість користувачів, кількість транзакцій) усіх користувачів послуг.

2.2 Модель багаторівневої хмарної мережі

Після наведеної проблематики варто дослідити модель багаторівневої хмарної мережі з наступними учасниками бізнес-процесу надання послуги:

– провайдери інфраструктури (IaaS-provider);

– провайдери платформи (PaaS-provider);

– провайдери ПО і готових рішень (SaaS-provider);

– кінцеві абоненти послуг.

Аналіз спирається на сучасні тенденції хмарних обчислень. В даний час проводиться безліч досліджень в сфері хмарних обчислень, і зокрема з питань оптимізації ресурсів[11]. Проте, на сьогоднішній день немає чіткого бачення,

яким чином різні учасники бізнес-процесу можуть співпрацювати з метою спільної вигоди.

Розглянемо багаторівневу хмарну архітектуру з наступними трьома рівнями віртуалізації:

– рівень IaaS - віртуалізація апаратного рівня і різних обчислювальних ресурсів (процесор, оперативна пам'ять, дисковий простір). Ресурси виділяються на вимогу і оплачуються за часом споживання;

– рівень PaaS - розробникам пропонується середовище для розробки, тестування та розгортання свої додатків, яка може включати різне базове ПО, засоби горизонтального і вертикального масштабування, можливості балансування навантаження та ін. Рівень PaaS використовує віртуальні ресурси і засоби управління ресурсами рівня IaaS. На одному рівні віртуалізації IaaS може бути розгорнуто кілька рішень PaaS;

– рівень SaaS складається з простих або композитних послуг, які пропонуються кінцевим абонентам. У моделі дане програмне забезпечення розгортається на базі платформ PaaS.

Кожен рівень у відповідності з основними властивостями хмарних обчислень включає засоби управління наданими ресурсами і програмними рішеннями. Розглянемо випадок, при якому рівні IaaS, PaaS і SaaS належать різним учасникам бізнес-процесу [12]. При цьому кожен рівень буде використовувати тільки ресурси і системи нижнього шару в режимі закритої архітектури.

В силу відмінності бізнес-цілей кожен шар буде мати різні цілі оптимізації, критерії та механізми.

Розглянемо основні цілі учасників бізнес-процесу надання послуги SaaS, а також доступні засоби оптимізації та критерії оптимізації. Схематично порівняльна характеристика зображена на рисунку 2.2.

У кожному шарі відображені основні економічні інтереси власника - збільшення прибутку або збільшення задоволеності кінцевих користувачів послуг[13].

Для власників ЦОД рівень доходів залежить багато в чому від кількості споживаних обчислювальних ресурсів і капітальних і операційних витрат на дані ресурси. У зв'язку з цим основним завданням оптимізації є мінімізація кількості обладнання при обслуговуванні запитів рівня PaaS в умовах забезпечення договірних зобов'язань на надання встановленої кількості мережевих і обчислювальних ресурсів.

Рівень IaaS має засоби моніторингу інформації про завантаження, наявність і розташування ресурсів, як для апаратного забезпечення, так і для віртуальних машин (ВМ). Основними засобами оптимізації рівня IaaS є перерозподіл обчислювальних потужностей і дискового простору для ВМ.

Рисунок 2.2 – Бізнес-цілі учасників процесу надання послуги SaaS

Доходи рівня PaaS пов'язані з системами та послугами SaaS, які були розміщені розробниками: споживані ресурси і ліцензії на програмне забезпечення. Провайдер PaaS оплачує ресурси, надані рівнем IaaS (віртуальні машини, дисковий простір, мережі передачі даних), вартість ліцензій третіх сторін (наприклад, на сервери програмного забезпечення або бази даних) і порушення угод про рівень обслуговування (SLA).

У зв'язку з цим, основними бізнес-цілями рівня PaaS є:

– збільшення навантаження на платформу з боку розробників додатків SaaS;

– мінімізація використовуваних ресурсів;

– мінімізація штрафів

– виконання умов SLA.

На рівні PaaS доступна можливість моніторингу ресурсів ВМ, контейнерів для програмне забезпечення SaaS і використовуваних ліцензій третіх сторін. Основними засобами оптимізації є наступні операції:

– зміна кількості, розміру і типу ВМ;

– розподіл контейнерів в ВМ.

Для власників SaaS рівень доходів, як правило, залежить від числа кінцевих користувачів і/або використовуваних разових можливостей додатку, наприклад, посекундна тарифікація послуг телефонного зв'язку в рамках віртуальної IP-АТС. Витрати, в свою чергу, включають оплату ресурсів і можливостей рівня PaaS (за обсягом використання або згідно підписці).

На рівні SaaS доступна можливість контролю якості надання послуг (QoS) з метою оптимізації відповідно до власних задач. Залежно від типу послуг, що надаються, можуть використовуватися різні параметри QoS, наприклад, час відгуку для запиту користувача [14]. При цьому для власників програмного забезпечення SaaS основними засобами оптимізації є:

– контроль кількості і якості ресурсів, що надаються PaaS;

– зміни топології рішення - настройка контейнерів розгортання додатків;

– налаштування параметрів програмного забезпечення;

– дослідження ПЗ;

– пошук вузьких місць у власному додатку, що вимагають багато ресурсів.

В результаті вивчення цілей, критеріїв і засобів оптимізації можна виділити загальний момент, єдиний для всіх учасників бізнес-процесу - це підтримка заданого рівня якості наданої послуги.

В хмарі, що знаходиться в процесі експлуатації, процес оптимізації виконується безперервно [15]. Це пов'язано з постійними змінами умов навантаження на всі рівні моделі хмарних обчислень:

– впроваджуються нові послуги і платформи;
– змінюється абонентська база послуг;
– виходять з ладу апаратні і програмні компоненти;
– проводяться заходи по обслуговуванню та ін.

Для рівнів IaaS і PaaS рішення по оптимізації здійснюється автоматично на базі поточного стану хмари, що включає інформацію про програмне забезпечення і виділених ресурсів і про сценарії оптимізації, які відстежують продуктивність кожного додатку в хмарі. В даний час подібні алгоритми вже розроблені і використовуються на практиці [14].

Для рівня SaaS питання оптимізації використовуваних хмарних ресурсів дещо складніше в силу наступних факторів:

– невеликі терміни розробки;
– в програмний код програмне рішення SaaS повинні бути закладені можливості оптимізації ресурсів відповідно до вимог рівня PaaS (інтерфейси управління ресурсами різні для кожної платформи PaaS);
– для розробників SaaS рішень питання оптимізації є другорядним завданням: особливо це актуально для стартапів, у яких відсутні необхідні ресурси;
– підходи до реалізації ПО є унікальними для всі додатків: важко застосувати єдині правила для оптимізації ресурсів хмарних обчислень.

У цій роботі досліджується імітаційна модель, що дозволяє розробникам SaaS рішень спростити процес контролю і оптимізації використовуваних ресурсів. Імітаційна модель зображена на рисунку 2.3.

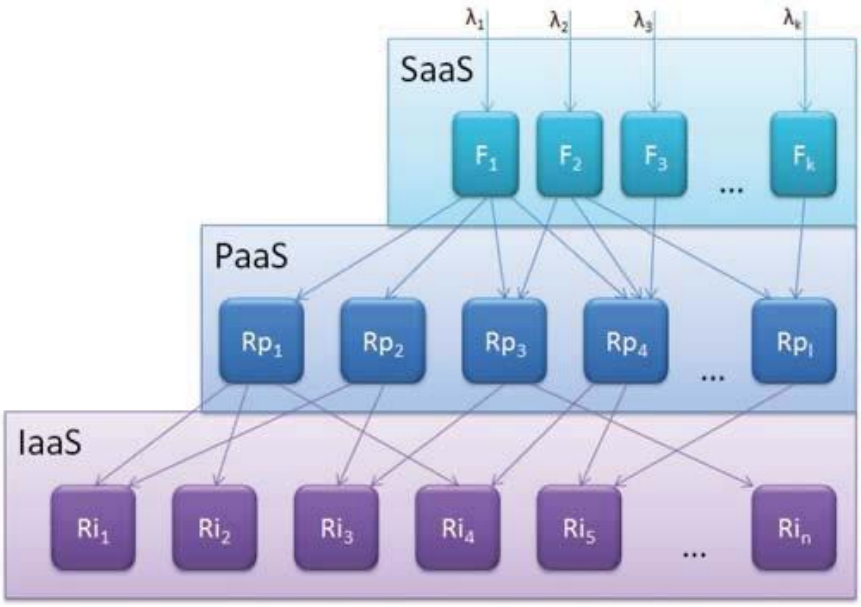

Рисунок 2.3 – Імітаційна модель контролю і оптимізації ресурсів хмарних обчислень

У моделі передбачається, що кожне програмне забезпечення за моделлю SaaS можна умовно розбити на k незалежних функцій F_i ,i=1,k. Припустимо, що виконання кожної функції F_i за запитом кінцевого користувача послуги SaaS зажадає певну кількість ресурсів рівня PaaS $(Rp_i,\ i = 1,\ l)$ протягом заданих інтервалів часу (Tf_{1i},1,l).

У даній моделі під ресурсами маються на увазі віртуальні обчислювальні потужності, які надаються контейнеру на рівні PaaS, наприклад:

– процесорні ресурси різних платформ за запитом (даний ресурс має фізичні обмеження для одного контейнера);

– об’єм пам'яті різних платформ за запитом (даний ресурс має фізичні обмеження для одного контейнера);

– дисковий простір і дискові операції для СУБД і звичайних сховищ;

– мережеві ресурси за запитом (внутрішні та зовнішні) і ін.

Окремо можна виділити ліцензії на програмні платформи рівня PaaS, які будуть враховуватися в залежності від використовуваної топології програм SaaS. Топологія програмного забезпечення наведена на рисунку 2.4.

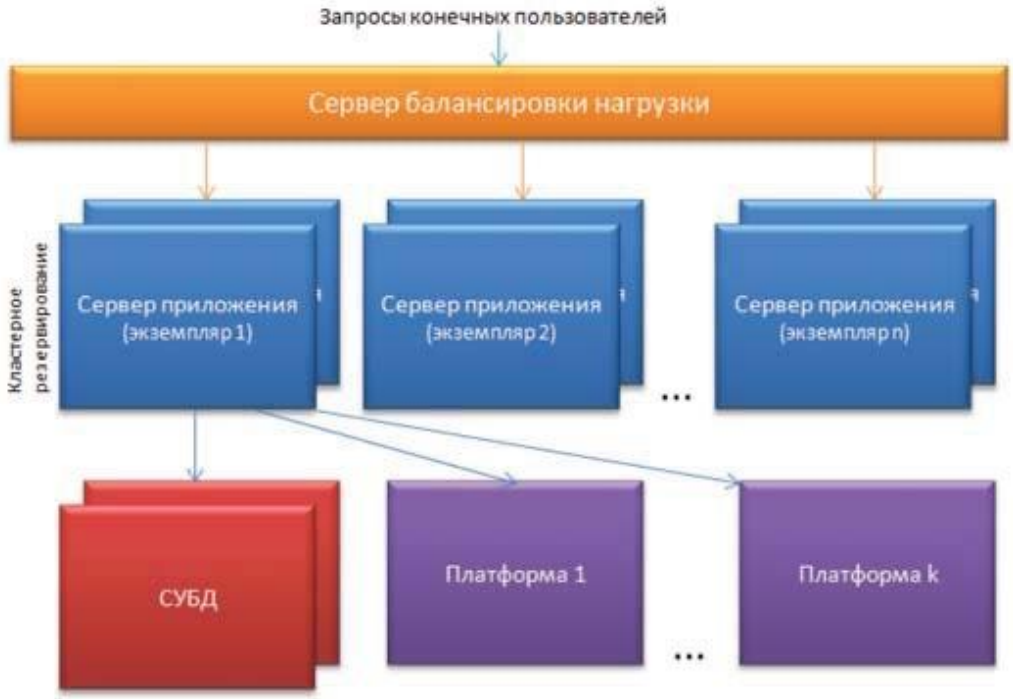

Рисунок 2.4 – Топологія програмних рішень за моделлю SaaS

Визначивши частоту надходження заявок на виконання функцій програми SaaS $\lambda_i, l=1,k,$ можна провести моделювання навантаження і досліджувати використовувані хмарні ресурси з точки зору:

– відображення використовуваних ресурсів для кожної окремої функції програми SaaS;

– відображення сумарного навантаження на кожен ресурс усіма функції програми SaaS;

– перевищення встановлених обмежень щодо використання ресурсу PaaS (потрібне коригування топології програмного забезпечення) і ін.

При розробці ПО даної імітаційної моделі додатково реалізуються наступні можливості:

– визначення параметрів потоку заявок на виконання функцій програми (з урахуванням години найбільшого навантаження);

– визначення топології програмного забезпечення SaaS;

– визначення вартості використання ресурсів PaaS;

– графічна візуалізація.

Надалі в імітаційної моделі планується додати такі можливості:

– обмежують ресурси на рівні IaaS (наприклад, канали телефонного зв'язку або ліцензії на ПЗ);

– можливість моделювання навантаження для компаній, що реалізують одночасно два рівня віртуалізації - SaaS і PaaS;

– облік різних залежностей використання ресурсів для функцій програми SaaS.

Як висновок сформулюємо основні підходи при розробці ПЗ SaaS, яких рекомендується дотримуватися для успішного просування власного рішення:

– програмне забезпечення має підтримувати можливість надання послуг великій кількості (тисячам) абонентів;

– при розробці розподіленої системи треба враховувати вибір регіону з розташуванням дата-центру;

– програмне забезпечення має забезпечувати засоби поділу даних абонентів (включаючи приховування даних від адміністраторів програмного забезпечення);

– програмне забезпечення має підтримувати можливості швидкого масштабування;

– при розробці і розміщенні програмного забезпечення має бути вирішено питання доступності до послуги відповідно до вимог SLA;

– при розробці програмного забезпечення повинна враховуватися швидкість виконання кожної транзакції;

– програмне забезпечення має надавати можливість кожному абоненту налаштовувати послугу відповідно до власних виробничих завдань, не змінюючи налаштування інших абонентів;

– програмне забезпечення має допускати глобальне використання без географічних обмежень, включаючи мовні.

Виходячи з наведених основних підходів, можна сказати що при розробці високорівневої платформи представлено вирішення ряду технічних та принципових питань взаємодії з розробниками сервісів та їх користувачами.

2.3 Порівняння сервісів хмарних обчислень за різними моделями

З наведених вище фактів та проблем у хмарному світі з’являється декілька основних підходів до створення будь-якого програмного рішення. У цій роботі буде розглянуто саме головних представників кожної моделі хмарного обчислення. Так як вище було наведено саме проблематику підходів IaaS, PaaS та SaaS, а для розробників програмного забезпечення використання програмного продукту як сервісу немає відношення, то варто також розгляну новий сучасний підхід до хмари FaaS.

Функція як сервіс або FaaS (Function as a service) це відносно новий термін, яким визначається можливість безсерверного запуску ділянок коду, що дає можливість розробникам писати і оновлювати ці ділянки коду «на льоту», які будуть запускатися в результаті відгуку на якусь подію, до наприклад, коли користувач натисне на елемент в веб-додатку. Завдяки цьому масштабувати код і впроваджувати мікросервіси стає на порядок простіше. По суті, FaaS - це просте поняття, яке означає, що:

– команда розробників не повинна турбуватися щодо таких аспектів, як бекенд-сервер, його обслуговування, закупівля чи масштабування (ну, певною мірою). Все, про що має турбуватися команда, - це логіка застосування;

– обробка проводиться на обчислювальних контейнерах, які не мають статусу;

– замість того, щоб мати сервер із тривалими запущеними процесами, скажімо «cron», тут обробка розпочинається лише після того, як відбудеться кваліфікована "подія" і припиняється, коли обробка завершена або встановлений час минув.

Після того, як термін кожної з моделей хмарних обчислень які можна віднести до Cloud обчислень (рисунок 2.5) було чітко визначено, стає можливим виділити представників кожної з цих моделей.

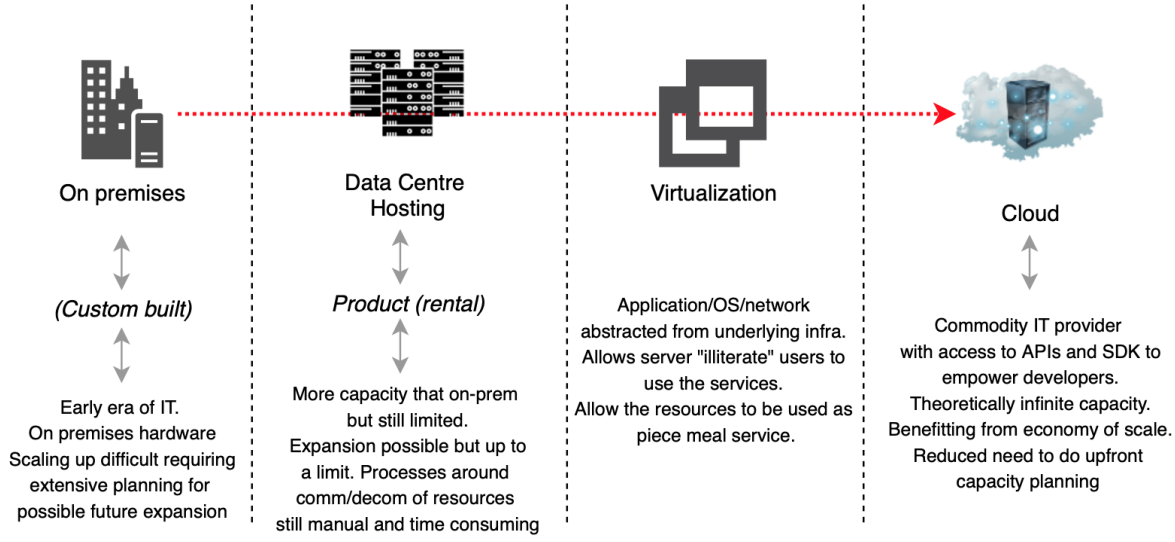

Рисунок 2.5 – Етапи становлення хмарних обчислень

Важливим моментом у проектуванні будь-якої «продакшн»-системи є, в першу чергу, вибір постачальника хмарних обчислень та послуг. У хмарних обчисленнях постачальники повинні працювати разом з клієнтом в напрямку створення довіри через співпрацю в розробці та налаштування політики безпеки, договорів і угод про рівень обслуговування, в питаннях законодавства і відповідності стандартам [16].

Тому, агрегуючи усі отримані дослідні та статистичні дані, дослідження будуть проводитись за допомогою сервісів Amazon.

Amazon має лідерство у великій «Хмарній війні». Оскільки міграція до хмари – це останній статус-кво в галузі програмного забезпечення, такі гіганти, як Amazon, Microsoft та Google, все ж змагаються на перше місце. Наразі, схоже, AWS виграє битву за хмарні обчислення, консолідувавши найбільшу частку ринку згідно з останніми аналітичними даними компанії Gartner у 2020-му році. Ці статистичні дані виведено у окрему діаграму на рисунку 2.6

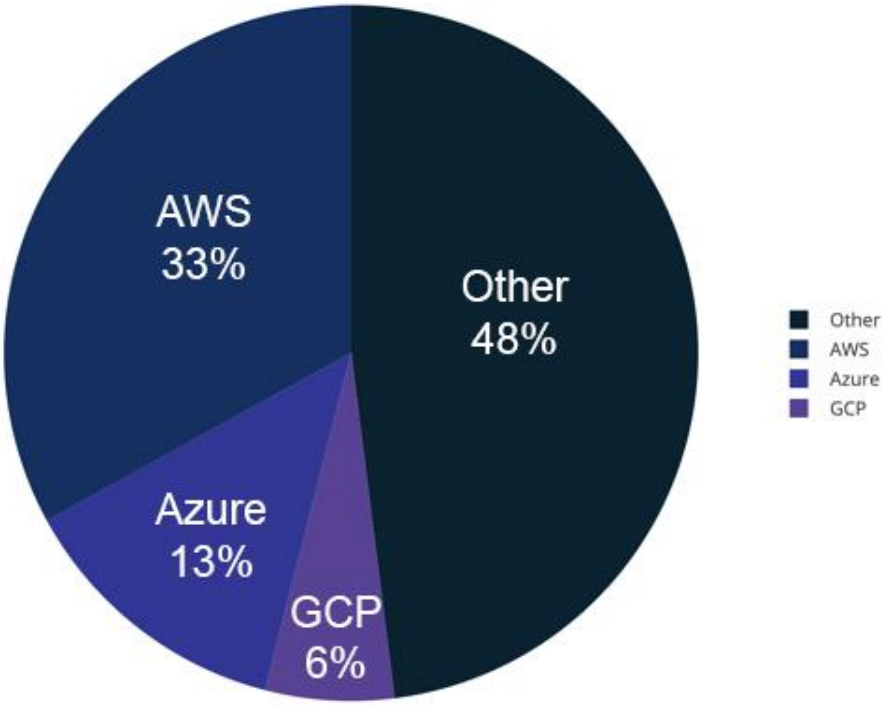

Рисунок 2.6 – Розподіл ринку між постачальниками хмарних обчислень

Amazon Web Services (AWS) – це найпоширеніша в світі хмарна платформа з широкими можливостями, що надає більше 175 повнофункціональних сервісів для центрів обробки даних по всій планеті. Мільйони клієнтів, в тому числі стартапи, які стали лідерами за швидкістю зростання, найбільші корпорації і передові урядові установи, використовують AWS для зниження витрат, підвищення гнучкості і прискореного впровадження інновацій. AWS надає незрівнянно більше сервісів і їх функцій, ніж будь-який інший постачальник хмарних послуг: від інфраструктурних технологій, таких як інструменти для обчислення, сховища і бази даних, до інновацій, наприклад машинного навчання і штучного інтелекту, озер даних і аналітики, а також Інтернету речей. З ними клієнт зможе швидше, легше і дешевше перенести поточні дані в хмару і реалізовувати в ньому будь-які можливі проекти [17].

Однак вибір постачальника, щоб перейти до хмари, – це лише половина битви. Інша половина передбачає визначення того, який сервіс найкраще

відповідає кожній конкретній система. Варіанти варіюються від інфраструктури як послуга (IaaS) до функцій як послуга (FaaS), з різними парадигмами між тим, що пропонують різні рівні абстракції. На рисунку 2.6 зображені сервіси, які відносяться до кожної моделі хмарних обчислень.

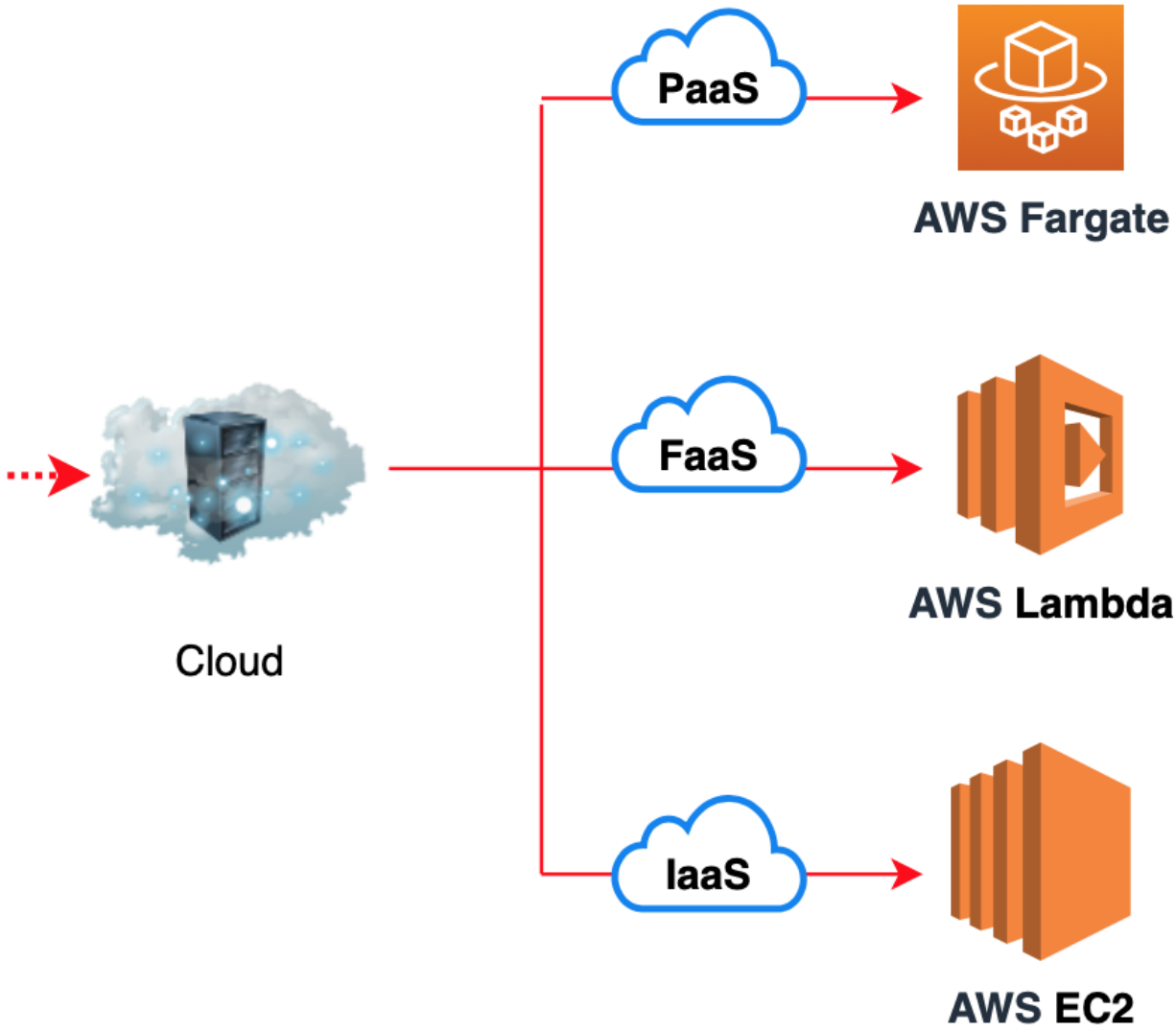

Рисунок 2.6 – Основні технології для підходів IaaS, PaaS та FaaS

Після ознайомлення з усіма моделями хмарних обчислень стає можливим більш детально переглянути інформацію про рішення в компанії Amazon.

2.3.1 AWS Lambda

AWS Lambda дозволяє запускати програмний код без виділення серверів і керування ними. Сплаті підлягає тільки фактичний час виконання обчислень.

За допомогою Lambda можна запускати практично будь-які види програмних сервісів та серверних рішень, при цьому не потрібні будь-які операції адміністрування. Просто завантажте програмний код, і Lambda забезпечить всі ресурси, необхідні для його виконання, масштабування і забезпечення високої доступності. Можна налаштувати автоматичний запуск програмного коду з інших сервісів AWS або викликати його безпосередньо з будь-якого мобільного або інтернет-додатки. AWS Lambda дозволяє налаштувати виконання програмного коду у відповідь на такі тригери, як зміна в даних або стан системи, а також певні дії користувачів. Функції Lambda можна безпосередньо запускати з інших сервісів AWS, наприклад S3, DynamoDB, Kinesis, SNS і CloudWatch, або вбудовувати в робочі процеси за допомогою AWS Step Functions. Це дозволяє створювати різноманітні бессерверной системи обробки даних, що працюють в режимі реального часу [18]. Принцип роботи сервісу AWS Lambda зображений нижче на рисунку 2.8.

Рисунок 2.8 – Принцип роботи сервісу AWS Lambda

Переваги та недоліки сервісу AWS Lambda наведено нижче у таблиці 2.1.

Таблиця 2.1 – Переваги та недоліки використання сервісу AWS Lambda

Риса	Коротка характеристика	Інформація
Перевага	Відсутнє управління сервером	AWS Lambda дозволяє автоматично запускати програмні коди без необхідності у виділенні серверів або управлінні ними. Достатньо написати програмний код і завантажити його в Lambda.
Перевага	Безперервне масштабування	AWS Lambda автоматично масштабує додаток, запускаючи програмний код у відповідь на кожен тригер. Всі запущені коди виконуються паралельно один до
Перевага	Безперервне масштабування	одного, при цьому кожен тригер обробляється в індивідуальному порядку, що забезпечує масштабування відповідно до робочої навантаженням.
Перевага	Оплата з точністю до часткою секунди	При роботі з AWS Lambda оплачуються кожні 100 мс виконання програмного коду і кількість його тригерів. Сплаті підлягає тільки фактичний час виконання обчислень.
Перевага	Висока продуктивність	З AWS Lambda ви можете оптимізувати час виконання коду, вибравши правильний обсяг пам'яті для вашої функції. Ви також можете включити Provisioned Concurrency, щоб ваші функції були ініційовані і готові до швидкого реагування в межах ста мілісекунд.
Недолік	Складні схеми викликів	Функції AWS Lambda обмежені часом виконання до 15 хвили. Це означає, що вам потрібно приділяти більше часу оркеструванню та організації своїх функцій, щоб вони могли поширюватися на ваші дані.

Недолік	Відсутній контроль над навколишнім середовищем системи	Функції AWS Lambda виконуються на машинних інстанціях Amazon (AMI) і, таким чином, використовують стандартні інструменти загальної розробки, ви не в змозі встановити спеціальні пакети чи програмне забезпечення на робочому середовищі.

Таким чином розробка додатків без серверів несе в собі багато компромісів. Для деяких випадків використання швидкість виконання є першорядною, а час, втрачений на обслуговування, - критичний час, який не можна собі дозволити. Функції AWS Lambda надають вам можливість використовувати цей недолік, дозволяючи від'єднати код програми від архітектури вашого сервера та створити більш еластичний, дешевший додаток, при цьому приносячи деякий контроль над середовищем виконання та тривалістю виконання.

2.3.2 AWS Fargate

AWS Fargate - це програмне ядро для безсерверних обчислень на базі контейнерів, яке працює як з Amazon Elastic Container Service (ECS), так і з Amazon Elastic Kubernetes Service (EKS). Fargate дозволяє повністю концентруватися на створенні додатків. При використанні Fargate не доводиться виділяти сервери і управляти ними. Крім того, Fargate дозволяє вказувати ресурси і оплачувати їх окремо для кожної програми, а вбудована ізоляція додатків підвищує рівень безпеки при роботі. Принцип роботи сервісу зображений нижче на рисунку 2.9.

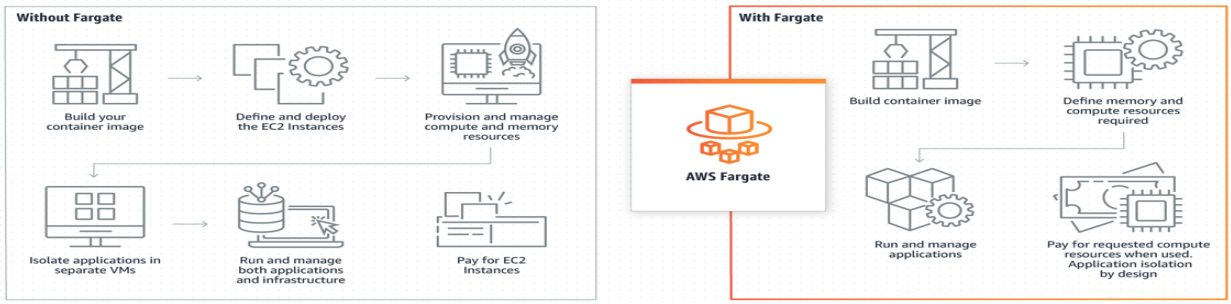

Рисунок 2.9 – Принцип роботи сервісу AWS Fargate

Переваги та недоліки сервісу AWS Fargate приведено нижче у таблиці 2.2.

Таблиця 2.2 – Переваги та недоліки використання сервісу AWS Fargate

Риса	Коротка характеристика	Інформація
Перевага	Розгортання додатків і управління ними, а не інфраструктурою	Завдяки Fargate ви зможете зосередитися на створенні і експлуатації додатків незалежно від того, де ви їх запускаєте - на ECS або EKS. Користувач взаємодіє тільки з контейнерами і сплачує тільки за них, що дозволяє уникнути операційних витрат на масштабування, установку виправлень і забезпечення безпеки серверів, а також на управління ними. Fargate забезпечує актуальність інфраструктури, в якій запущені ваші контейнери, за рахунок своєчасної установки необхідних виправлень.
Перевага	Правильний підбір ресурсів і гнучке ціноутворення	Fargate запускає і масштабує обчислювальні ресурси таким чином, щоб вони точно відповідали вимогам до ресурсів, зазначеним для контейнера. При використанні Fargate не потрібно надлишкове виділення ресурсів і оплата додаткових серверів. Як і при використанні інстанси Amazon EC2, при роботі з Fargate можна вибрати як спотові ціни, так і Compute Savings Plan. У порівнянні з цінами на вимогу Fargate надає знижку до 70% на додатки.
Перевага	Правильний підбір ресурсів і гнучке ціноутворення	Пропонується знижка до 50% на витрати, пов'язані з постійними робочими навантаженнями.
Перевага	Надійна вбудована ізоляція	Окремі завдання ECS або модулі EKS запускаються у власних середовищах виконання на виділених ядрах і не ділять ресурси ЦП, пам'яті, сховища і мережі з іншими завданнями і модулями. За рахунок цього досягається ізоляція робочих навантажень і підвищується рівень безпеки кожного завдання або модуля.
Перевага	Докладні дані про відстеження додатків	При роботі з AWS Lambda оплачуються кожні 100 мс виконання програмного коду і кількість його тригерів. Сплаті підлягає тільки фактичний час виконання обчислень.
Перевага	Висока продуктивність	У сервісі Fargate передбачено відстеження додатків за рахунок вбудованої інтеграції з іншими сервісами AWS, в тому числі Amazon CloudWatch Container Insights. Fargate пропонує широкий вибір інструментів сторонніх розробників з відкритими інтерфейсами, які дозволяють збирати метрики і журнали для відстеження додатків.
Недолік	Менше налаштувань	Недоліком Fargate є те, що користувач жертвує параметрами налаштування для

		зручності використання. Як результат, Fargate недостатньо підходить для користувачів, які потребують більшого контролю над своїми контейнерами. Ці користувачі можуть мати особливі вимоги до управління, управління ризиками та дотримання вимог, які потребують тонкого налаштування контролю над їх IT-інфраструктурою.
Недолік	Більше фінансових витрат	Fargate - це можливість економії коштів у правильній ситуації при переході від ECS або EKS. Однак для простих випадків використання Fargate може виявитися дорожчим. Amazon стягує з користувачів Fargate вищу плату за годину, ніж користувачі ECS та EKS. Це компенсує складність управління інфраструктурою ваших контейнерів.

Fargate виділяє необхідну кількість обчислювальних ресурсів, завдяки чому не потрібно вибирати інстанси і масштабувати ресурси кластера. Ви оплачуєте тільки ресурси, необхідні для запуску контейнерів. Вам не потрібно турбуватися про надмірне виділення ресурсів і оплати додаткових серверів. Fargate запускає кожен модуль або завдання на окремому ядрі, надаючи їм власну ізольовану обчислювальну середу. Завдяки цьому додаток отримує вбудовану ізоляцію робочих навантажень і поліпшену безпеку. Саме тому такі клієнти, як Vanguard, Accenture, Foursquare і Ancestry, вибрали для запуску своїх критично важливих додатків саме Fargate [19].

AWS Fargate все ще є новою технологією, але вона вже отримала в основному позитивні відгуки про платформу технологічного огляду G2 Crowd. Станом на момент написання цієї роботи, AWS Fargate отримав середній бал 4,5 з 5 зірок від користувачів G2 Crowd. Але не зважаючи на ці переваги, у клієнтів AWS Fargate є скарги.

2.3.3 AWS EC2

Обчислювальний хмара Amazon Elastic Compute Cloud (Amazon EC2) - це веб-сервіс, що надає безпечні масштабовані обчислювальні ресурси в хмарі. Він допомагає розробникам, спрощуючи проведення обчислень в хмарі в масштабі всього Інтернету. Використання Amazon EC2 позбавляє

користувачів від необхідності інвестувати кошти в апаратне забезпечення заздалегідь, щоб вони могли швидше розробляти та розгортати додатки.. Користувачі можуть використовувати Amazon EC2 для запуску стількох віртуальних серверів, скільки вам потрібно, обирати налаштування безпеки та мережі, а також управляти сховищем. Amazon EC2 дозволяє користувачам масштабувати інфраструктуру, щоб впоратися зі змінами в потребах або різкого збільшення потоку клієнтів [20].

Переваги та недоліки сервісу AWS EC2приведено нижче у таблиці 2.3.

Таблиця 2.3 – Переваги та недоліки використання сервісу AWS EC2

Риса	Коротка характеристика	Інформація
Перевага	Просте масштабування ресурсів	EC2 має широкий спектр конфігурацій машин. Якщо заплановані рішення важкі для пам'яті, важкі для процесора, важкі для GPU або IO важкі, EC2 може надати належну конфігурацію машини відповідно до вимог.
Перевага	Правильний підбір ресурсів і гнучке ціноутворення	EC2 має багато машинних образів ОС та необхідних програмних засобів. Також EC2 дозволяє створити образ власного диску. Це полегшує користувачу зупинити екземпляр EC2, не втрачаючи роботоспроможність системи. Образи можна знову приєднати до EC2, щоб почати з того самого місця, де воно було залишене.
Перевага	Широкий спектр варіантів оренди екземплярів EC2	Amazon дозволяє по-різному купувати екземпляри EC2: «on-demand», «spot» та «reserved». Залежно від потреби, можна прийняти правильне рішення, щоб заощадити витрати.
Недолік	Високий поріг входження у технологію	Новим користувачам потрібний великий проміжок часу для навчання, щоб ефективно використовувати цей сервіс.
Недолік	Складний UI у веб-консолі	Користувацький інтерфейс для служби EC2 є складним і він у багатьох місцях пропускає детальне пояснення функціоналу.
Недолік	Неочікуваний старт функції автомасштабування	Іноді дуже важко зрозуміти чому екземпляр автоматично масштабується, коли ви користувач впевнений що в цей конкретний час немає великого трафіку.

Основними характеристиками сервісу EC2 є:

– інтеграція: EC2 може інтегруватися з іншими службами AWS, такими як RDS, SimpleDB та SQS.

– точний контроль: користувачі отримують адміністративний доступ до своїх екземплярів, можуть зупиняти і запускати екземпляри, зберігаючи дані завантажувального розділу та можуть отримати доступ до консольного виводу для цього примірника.

– безпека: користувачі можуть контролювати, які екземпляри залишаються приватними та які мають Інтернет. EC2 використовує для безпеки Virtual Private Cloud (VPC) Amazon, і підприємства можуть підключати свою захищену IT-інфраструктуру до ресурсів VPC.

– вартість: серед ряду варіантів ціноутворення EC2 пропонує доступні погодинні ставки;

– наслідування: DevOps команді робота з сервісом EC2 має бути знайомою, адже це нічим не відрізняється за кроками від базової конфігурації серверу.

Існує кілька типів екземплярів AWS з різною конфігурацією та перевагами, які відрізняються не тільки характеристиками, але й ціною:

– для загальних цілей (General purpose);

– оптимізовані для обчислень (Compute optimized);

– оптимізовані під використання ОЗУ (Memory optimized);

– прискорені обчислення (Memory optimized);

– оптимізовані для зберігання даних (Storage optimized).

Слово "еластичний" в Elastic Compute Cloud говорить про здатність системи пристосовуватися до різного навантаження та резервування ресурсів або припинення ресурсів відповідно до попиту.

3 ПОРІВНЯННЯ СЕРВІСІВ ХМАРНИХ ОБЧИСЛЕНЬ ЗА РІЗНИМИ МОДЕЛЯМИ

3.1 Введення у способи порівняння

Вибір серед сервісів описаних в попередньому розділі включає два основні заходи, які в кінцевому підсумку диктують життєздатність вашої програми в хмарі. Сюди входить операційне навантаження роботи на хмарі та витрати у розрізі жертв технологіями заради економії на проекті. Отже, вибір правильної послуги означає збалансування двох заходів для досягнення оптимального досвіду, і цей акт балансування виконується відповідно до обраної вами хмарної парадигми.

В даний час AWS пропонує обчислювальні послуги у формі IaaS з відомим контейнерним сервісом EC2, FaaS у вигляді безсерверних функцій Lambda і PaaS з нещодавно випущеним продуктом Fargate. Плюси і мінуси цих різних продуктів можна знайти в різних блогах та на форумах, але вони не відповідають безпосередньо, яку послугу слід використовувати конкретно для розробки різного ПЗ. Таким чином, саме ця стаття допоможе вам вирішити, яка з трьох основних служб хмарних обчислень AWS найкраще відповідатиме вашому конкретному випадку використання відповідно до двох згаданих заходів.

3.2 Порівняння операційної частини

Одне з привабливих обіцянок хмарних обчислень - це істотне зниження складності управління апаратними засобами сервера. З зростанням пропозицій IaaS люди могли просто делегувати обов'язки управління обладнанням далеко від постачальників хмарних технологій. Однак розробникам тепер довелося

навчитися надавати віртуальним серверам через хмарні платформи постачальників, вводячи новий тип операційного навантаження. З тих пір IaaS еволюціонував протягом багатьох років в Контейнери як послуга (CaaS) до Платформа як послуга (PaaS), нарешті, Функції як послуга (FaaS). Всі ці парадигми пропонують різні рівні абстракції, FaaS - це найпростіша парадигма для використання в хмарних обчисленнях, оскільки вона має найвищі рівні абстракції.

Як можна побачити з рисунку 3.1, є три обчислювальних сервіси на вибір у хмарному середовищі AWS, де EC2 має найбільше експлуатаційне навантаження, AWS Lambda - найменше, а AWS Fargate - десь посередині спектра.

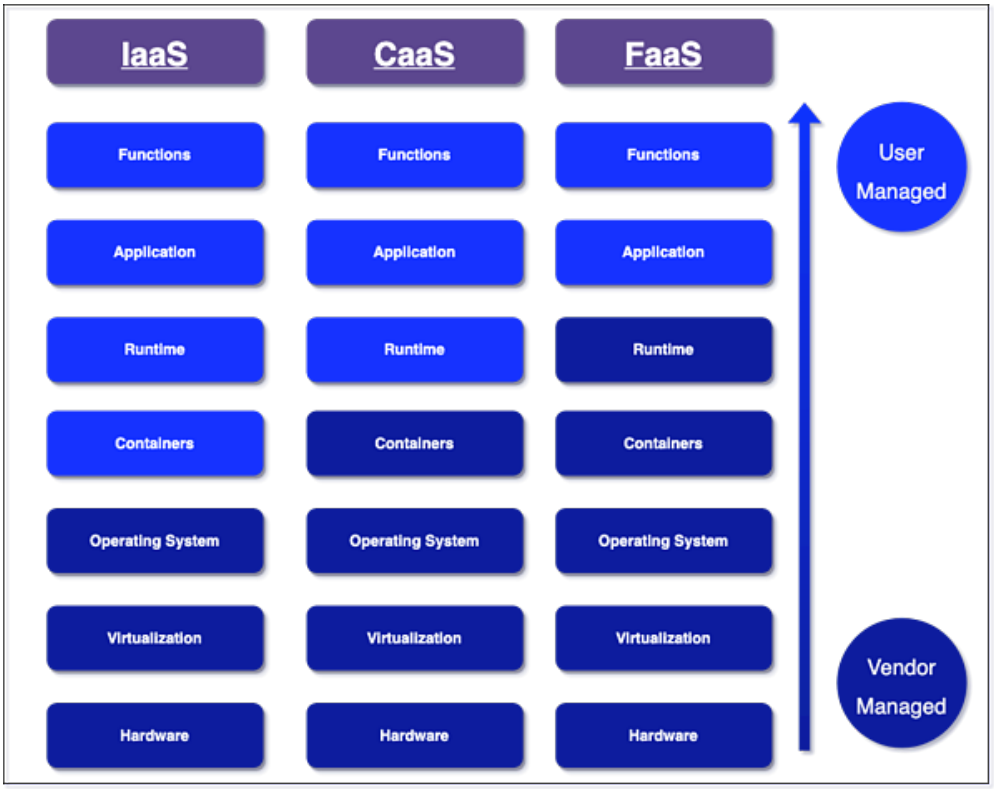

Рисунок 3.1 – Рівні абстракції між парадигмами

Однак є в цій справі є один трюк. Оскільки більша кількість абстракцій призводить до менших експлуатаційних навантажень, потрібно пожертвувати гнучкістю та мати справу з експлуатаційними обмеженнями. Наприклад, із EC2 екземпляром ви повинні вказати правила, моніторинг безпеки мережі та

багато іншого. Крім оркестрації контейнерів, головним питанням є також автоматичне масштабування, визначаючи правила масштабування на рівні контейнера як надзвичайно важкі. Все це додаткове навантаження в роботі означає, однак, що користувач може налаштувати своє оточення практично будь-яким способом, який буди доречний для конкретної системи. Таким чином, користувач можете вибрати будь-який час виконання, наприклад, не турбуватися про обмеження часу, а також визначити детальні правила автоматичного масштабування, які найкраще відповідають користувацьким потребам бізнесу.

З іншого боку, з AWS Lambda все це може бути або відібрано, або доповнене інструментами, такими як Thundra.io для моніторингу. Ще одна перевага полягає в тому, що сервіс працює надзвичайно добре, коли справа стосується автоматичного масштабування. Є невеликі недоліки щодо моніторингу, оскільки сервіс AWS Lambda відомий своєю нездатністю забезпечити суттєвий моніторинг, але за допомогою Thundra.io ви можете отримати повне спостереження за своїм діловим кодом за умови, що навіть перевершує моніторинг у двох інших службах AWS. Отже, з усіма цими інструментами SaaS та простотою експлуатації навколо AWS Lambda, використання сервісу дозволяє зосередити увагу насамперед на вашій бізнес-логіці. Однак тепер вам доведеться мати справу з експлуатаційними обмеженнями послуги.

AWS Lambda дійсно обмежує гнучкість роботи в порівнянні з EC2 з іншого боку спектра. Деякі з цих обмежень включають обмеження ефемерного простору на диску на 512 МБ та розмір пакета для розгортання на 50 Мб, та кілька інших. Отже, навіть якщо вам більше не доведеться турбуватися про надання контейнерів та визначення правил масштабування, вам все одно доведеться турбуватися про описані вище обмеження.

Однак AWS Fargate знаходиться між спектром EC2 та AWS Lambda на спектрі і забезпечує перепочинок від обмежень Lambda. Так само і є Fargate. Це залежить від того, наскільки простоти в роботі користувач хоче досягти.

Наприклад, з Fargate більше не доведеться турбуватися про виправлення, подібне до EC2, але все одно доведеться оновлювати базові контейнери. Крім того, як показано на рисунку 3.2, незважаючи на те, що за допомогою Fargate можливо врятуватися від оркестрації EC2, все одно доводиться реєструвати свої контейнери на службі ECS, оскільки Fargate управляється через ECS, що ще більше ускладнює експлуатацію.

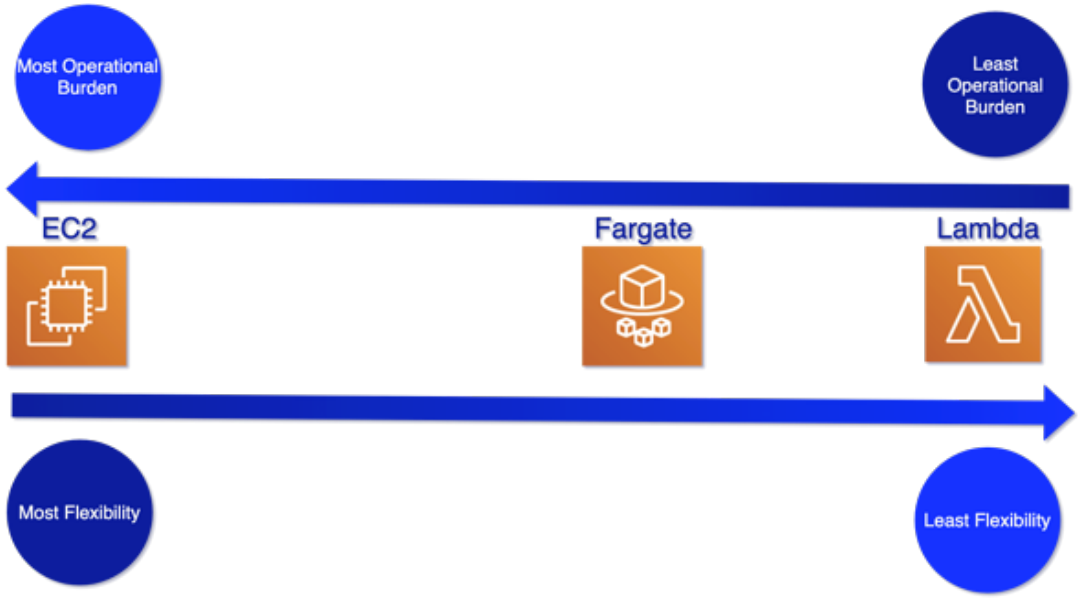

Рисунок 3.2 – Спектр управління операціями та гнучкістю послуг AWS

Тому, якщо мова йде про експлуатаційні витрати, є можливим або отримати повну гнучкість з EC2 або повну простоту експлуатації з AWS Lambda, або навіть досягти компромісу з Fargate. Але незалежно від того, що обирається відповідно до потреб для бізнесу, все одно доведеться платити за послугу, і саме тут входить другий параметр.

3.3 Порівняння фінансової частини

Міграція в хмару, зокрема, може призвести до надзвичайних фінансових витрат, або може заощадити багато грошей. Все залежить від обраної послуги

та способів використання цінових планів AWS. Тому необхідно, щоб користувачі зрозуміли, яка послуга найкраще відповідає конкретному випадку використання, особливо коли мова йде про оплату хмарних обчислень. Загальне правило полягає в тому, що чим більше рівень абстракції, тим дешевше послуга, але це не завжди має бути так.

По-перше, порівняння вартості AWS Lambda, екземпляра EC2 та AWS Fargate, не є простим завданням, оскільки перенесена вартість залежить від різних факторів. Наприклад, завдяки безсерверному характеру AWS Lambda і Fargate, сервіси дотримуються моделі оплати. AWS Lambda стягує плату за виклик та тривалість кожного виклику, тоді як AWS Fargate стягує з вас витрати на vCPU та ресурси пам'яті контейнерних програм, що використовуються в секунду.

І навпаки, виставлення рахунків для екземплярів EC2 залежить від того, який із чотирьох типів моделей ціноутворення користувач обрав. Сюди входить запит на замовлення, зарезервовані екземпляри, точкові екземпляри та виділені хости. Ознайомитись із політикою ціноутворення кожної моделі можливо перейшовши на сторінку цін EC2.

Отже видно, що порівняння різних служб не є простим завданням. Порівнюючи ціни, спочатку треба вирівняти умови гри, ігноруючи пропозиції щодо вільного рівня, і розглянемо випадки EC2 за запитом, щоб отримати витрати на годину. Крім того, ми оберемо достатньо потужний екземпляр m4.large, який включає 2 vCPU і пам'ять 8GiB, вартість яких становить 0,10 доларів на годину. Порівняно, AWS Fargate коштував би вам $ 0,11652 / годину за ті ж конфігурації, трохи дорожче, ніж екземпляр EC2.

На рисунку 3.3 показано порівняння між EC2 та Fargate за типами екземплярів, щоб краще зрозуміти, як відрізняються витрати між двома послугами.

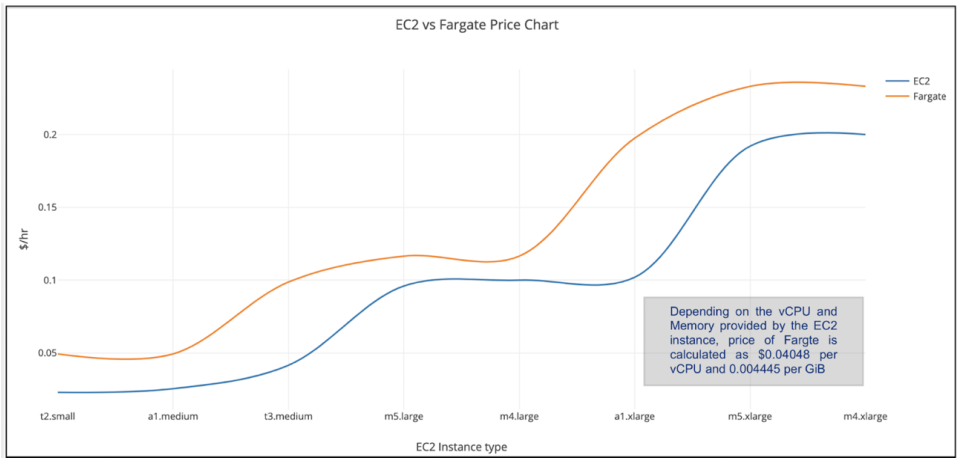

Рисунок 3.3 – Графік порівняння цін EC2 та Fargate

Треба мати на увазі, що за використання Fargate стягуватимете плату лише в тому випадку, якщо таск (завдання) виконується порівняно з фіксованою платою, що виникає при використанні сервісу EC2.

Тепер, розглядаючи AWS Lambda, потрібно думати з точки зору викликів кожної окремої функції. Це означає скільки закликів потрібно відповідно до поточної ціни виклику та тривалості протягом виконання програми щоб зрівнятись із вартістю екземпляра EC2. Оскільки для порівняння ми обираємо екземпляр m4.large EC2, рисунок 3.4 показує скільки викликів на годину нам потрібно буде виконати навіть залежно від різних розподілів пам'яті.

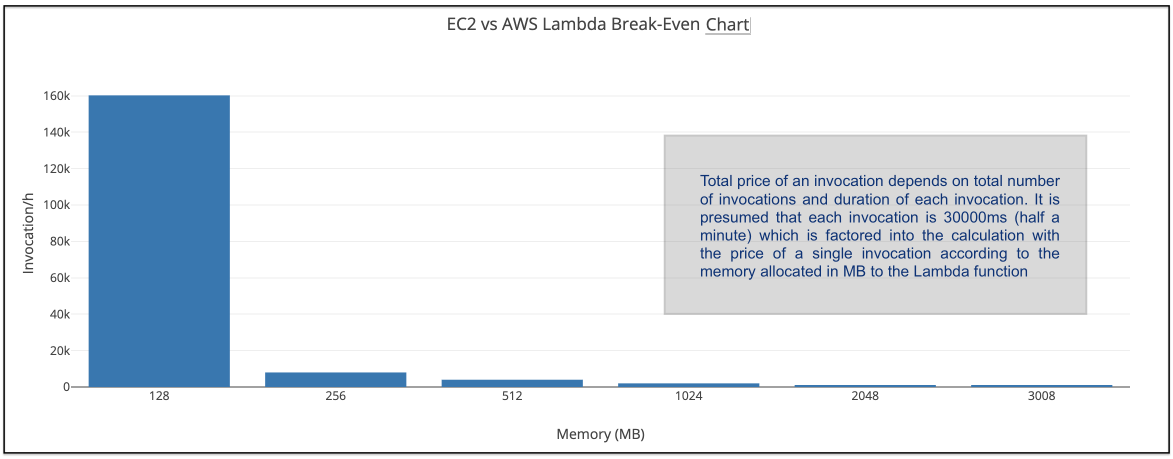

Рисунок 3.4 – Діаграма беззбитковості сервісу AWS Lambda

Як видно з діаграми, якщо тривалість кожного виклику вважається півхвилини, то користувач може домогтися приблизно 16000 викликів за годину, щоб зрівнятися з вартістю екземпляра m4.large EC2. Однак якщо користувач виділятиме максимальну пам'ять у 3008 МБ, тоді стане можливим досягти лише 660 викликів на годину.

Звідси видно, що рішення про вибір послуги AWS залежить від того, наскільки користувач хоче принести в жертву оперативність, і якщо вибір стане за AWS Lambda, скільки викликів очікується виконати.

3.3 Порівняння Serverless підходів

В останні роки Serverless набуло більшої популярності, ніж проста віртуалізація, оскільки інженерні організації ухилилися від складності та витрат, пов'язаних з керуванням фізичними серверами та навіть віртуальними машинами, розміщеними на хмарі, як AWS. Якщо роздивлятись Serverless на AWS, то слід порівнювати Lambda та ECS Fargate, щоб визначити, що найкраще відповідає використанню.

Lambda і Fargate принципово відрізняються як Serverless рішення, і важко зрозуміти цю різницю, переглядаючи лише сторінки маркетингу або документації AWS. Спочатку варто розібрати основи кожного сервісу щоб потім порівняти їх.

3.3.1 AWS Fargate та AWS Lambda у розробці ПЗ

Однією з областей, яку часто не помічають у порівнянні сервісів AWS Fargate та AWS Lambda є те, як працюють технології в розвитку. Ефективна

розробка програми так само важлива, як і ефективне її використання у виробництві, і девелопери завжди повинні враховувати як працюють різні рішення в середовищі розробки.

Першим розглянемо сервіс Fargate. Оскільки Fargate по суті просто працює з докерними контейнерами, ви можете розвивати локально, використовуючи Docker і запускаючи свої docker-образи в Fargate, не турбуючись про сумісність. Docker-compose - це чудовий інструмент розробки для локального запуску контейнерів докерів та має ті самі функції, які доступні для завдань ECS для налаштування мережі, вимог до ресурсів та легкого управління багатьма контейнерами докера. Розробка в docker вже дуже популярна, і підтримка спільноти широко доступна практично для будь-якого уявного сценарію. У цьому сенсі Fargate перевершує Lambda, оскільки архітектура та процес розвитку можуть бути повністю незалежними від Fargate.

Другим буде розглянуто сервіс Lambda. У перші дні сервісу AWS Lambda розробникам було дуже складно девелопити ПЗ, оскільки обмеження часу виконання та відсутність чіткого способу виконання функцій Lambda у розробці могли легко викликати проблеми залежності між розробкою та релізом у продакшн систему. Доводилося запускати функції dev Lambda у фактичному обліковому записі AWS, щоб точно знати, як це буде працювати, що було складно з командами розробників і вимагало від організації розробити інструменти для управління цим процесом. В останні роки як AWS, так і громада програмістів, розробили багато інструментів для полегшення створення, розгортання та управління функціями Lambda:

– інструменти для розробників AWS Serverless;
– покрокове налагодження запуску лямбда функцій локально;
– localStack - повністю функціональний локальний стек хмари AWS.

Навіть за допомогою цих інструментів розробка для Lambda може бути складним завданням для великих організацій і вимагатиме певних зусиль з боку команди розробки щоб стандартизувати які інструменти

використовуються і як виглядає процес розробки, забезпечення якості та розгортання функцій Lambda.

3.3.2 AWS Fargate та AWS Lambda у фінансових питаннях

Що стосується фінансових питань, то Lambda і Fargate використовують повністю окремі моделі виставлення рахунків, які важко порівняти один з одним або іншими параметрами для запуску коду. Більшості організацій доведеться вкласти великі кошти в оцінку витрат на Lambda та Fargate або просто спробувати їх обидва, щоб побачити різницю в реальних навантаженнях.

Першим буде розглянуто сервіс Fargate. Fargate виставляється на рахунок процесора та пам'яті, що використовується за годину. ECS не дозволяє налаштувати Fargate на використання довільних обсягів процесора та пам'яті, а обсяг пам'яті, що доступний, залежить від обсягу налаштованого процесора. Ідентично до Lambda, будь-який невикористаний процесор чи пам'ять, це, по суті, даремно витрачені гроші, тому потенційний користувач має розрахувати потреби свого програмного забезпечення відповідно.

Оскільки ціна на Fargate не ґрунтується на кількості запитів чи тривалості запиту, визначити нижню та верхню межі для використання, отримати гарне уявлення про діапазон цін, в якому буде знаходитись користувач є більш простою задачею. Можна просто порахувати кількість контейнерів які використовуються, порівняти їх із наявними конфігураціями процесора та пам'яті Fargate та обчислити щоденну вартість. Також є можливість використовувати показники CpuUtilized та MemoryUtilized, доступні в Container Insights, щоб допомогти розмістити свої завдання ECS у Fargate у продакшині.

Для передбачуваних навантажень на більш високих масштабах може бути доцільним навіть використовувати екземпляри EC2 для виконання деяких завдань ECS, а не покладатися виключно на Fargate. Ви втрачаєте "безсерверний" аспект виконання завдань ECS, але EC2 може бути значно дешевшим за допомогою точкових або зарезервованих екземплярів.

Наступним буде розглянуто сервіс Lambda. "Лямбда" оплачується за суму кількості запитів, пам'яті та секунд виконання функції. Намагаючись керувати витратами Lambda, важливо правильно розмістити вимоги пам'яті вашої функції, оскільки це безпосередньо впливає на вартість. Якщо користувач налаштує свою функцію на використання пам'яті 512 Мб, але вона споживає лише 128 Мб під час виконання, то істотно збільшується вартість кожного виклику в 4 рази. Lambda пропонує багато ярусів для використання пам’яті, тому не повинно бути надто складно знайти розмір, який повністю відповідає фактичному використанню.

Однією з порад по оптимізації витрат є можливість налаштувати CloudWatch Billing Alarms, щоб було можливим уникнути сюрпризів у своєму рахунку через Lambda. Завдяки цього стає можливим переглянути свої журнали функцій лямбда, щоб побачити використану пам'ять та час виконання ваших викликів лямбда, щоб допомогти користувачам правильно розмістити свої функції лямбда.

3.3.3 Моніторинг сервісів AWS Fargate та AWS Lambda

Моніторинг безсерверних програм може бути дуже складним. Традиційні засоби моніторингу серверів не дуже добре перекладаються на Lambda і Fargate, і при швидких темпах змін від розробників може бути дуже важко залишатися на вершині зростаючої інфраструктури без серверів.

Fargate експортує показники в CloudWatch автоматично. Є можливість переглянути ці публікації в блозі, де пояснюється, як налаштувати сигнали CloudWatch для використання процесора та пам'яті для ECS. Як і у Lambda,

більшість інструментів моніторингу та реєстрації підтримують ECS, а тому і Fargate. Також можна надсилати журнали до CloudWatch та інших постачальників із належною конфігурацією.

Інформація про контейнери забезпечує ще більшу видимість завдань ECS із покращеним веденням журналу та показниками. Хоча ця функція є лише у бета-версії, вона виглядає багатообіцяючою, що Amazon готовий докласти більше зусиль для підвищення видимості за допомогою ECS.

Важливі показники Lambda функцій експортуються до CloudWatch автоматично. Окрім того, що доступно у CloudWatch, існує безліч інструментів моніторингу та реєстрації даних, які підтримують Lambda, таких як Datadog, SignalFX, SumoLogic та багато іншого. Інструмент Blue Matador також автоматично відстежує Lambda та створює та керує сповіщеннями без конфігурації [22].

Одне завдання з моніторингу Lambda полягає в тому, що немає базового сервера, на якому можна запускати агенти моніторингу. Можливо здається необхідним передавати показники програми у межах своєї функції для задоволення потреб моніторингу, що може спричинити витрати як часу виконання, так і передачі даних.

4 ВИПРОБУВАННЯ ПРОДУКТИВНОСТІ SERVERLESS СЕРВІСІВ

4.1 Опис підходів на основі AWS Lambda та AWS Fargate

Користувач зробить HTTP POST-запит до кінцевої точки (ендпоінту), який буде оброблятися API Gateway сервісом. API Gateway направить запит на функцію AWS Lambda для обробки. Функція Lambda надсилає запит з корисним навантаженням на SNS topic перед поверненням відповіді. Код який буде використано у AWS Lambda наведений у додатку Б.

Схема сумісної роботи компонентів зображена на рисунку 4.1.

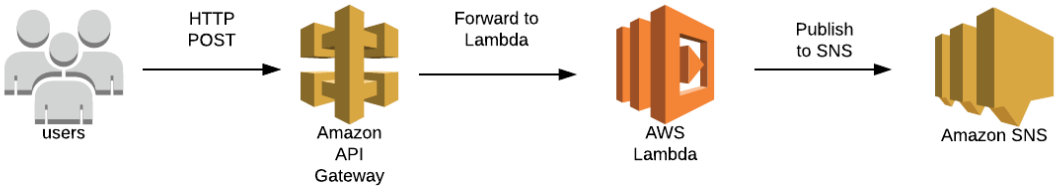

Рисунок 4.1 Схема взаємодії компонентів системи за участі AWS Lambda

Другий підхід - запустити обчислення в контейнерах Докера. Існує кілька різних підходів для цього на AWS, але я вирішив використовувати AWS Fargate. Схема сумісної роботи компонентів зображена на рисунку 4.2.

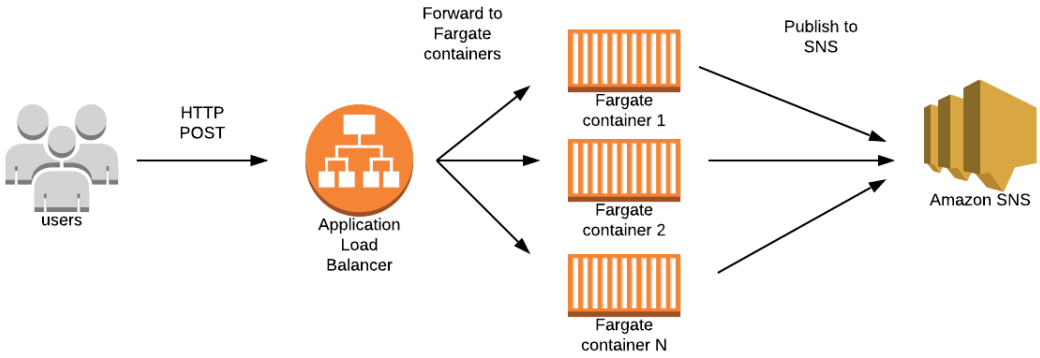

Рисунок 4.2 Схема взаємодії компонентів системи за участі AWS Fargate

Користувачі будуть надсилати запити HTTP POST до кінцевої точки HTTP, яка буде оброблятися балансиром завантаження додатків (ALB). Цей ALB буде пересилати запити до наших екземплярів контейнерів Fargate. Додаток на екземплярах контейнерів Fargate буде пересилати корисне навантаження запиту SNS.

За допомогою Fargate стає можливим запускати завдання чи послуги. Завдання - це разовий контейнер, який працюватиме до тих пір, поки він не загине або не закінчить виконання. Служба - це визначений набір певної кількості примірників завдання. Fargate забезпечує правильну кількість примірників вашої служби.

У цьому дослідженні буде використовуватись сервіс щоб мати змогу запустити достатню кількість екземплярів. Крім того, є можливість легко налаштувати балансир навантаження для управління HTTP-трафіком в усіх інстансах служби. Код та інструкції щодо розгортання цієї архітектури в Fargate можна знайти тут. Також буде використано інструмент CLI Fargate, який допомагає дуже просто перейти від Dockerfile до запущеного контейнера.

4.2 Результати тестування продуктивності

Два типи тестування продуктивності, які планується виконати після розгортання обох типів архітектури.

По-перше, оскільки обидві архітектури базуються на подіях, є сенс розігріти інфраструктуру: невелика партія з 2000 запитів, щоб дозволити службам запустити систему, яку було налаштовано раніше. Цей показник був близько 40 запитів в секунду.

Після цього решту робочого навантаження направили на інфраструктуру: 15000 запитів подивитися, як працює кожна архітектура, коли вони прогріваються.

Інтенсивність цієї частини тестування продуктивності становить 100 запитів в секунду. Для перших 2000 запитів до кожного типу кінцевих точок результати роботи такі:

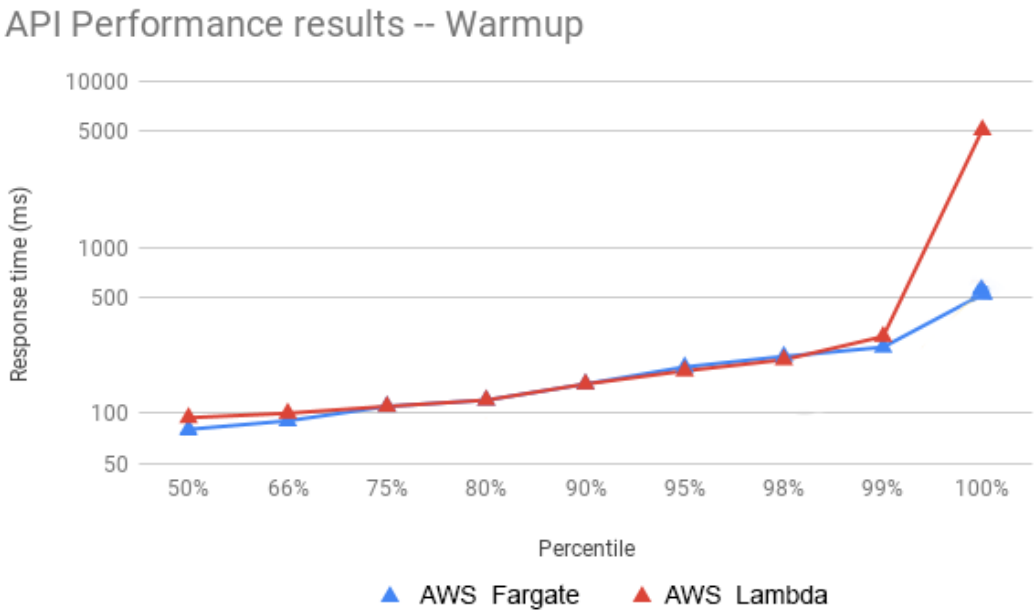

Рисунок 4.3 Графік тестування продуктивності API у режимі прогрівання

Виходячи з графіку зображеного на рисунку 4.3 можливо стверджувати що:

– Fargate стабільно був найшвидшим у всіх гранях дослідження.

– AWS Lambda мав найдовший хвіст у всіх. Це пов’язано з проблемою холодного пуску. Холодний старт викликаний початковим процесом створення контейнера, на якому працює завантажений код. Залежно від мови, на якій написане програмне рішення, затримка часто може перевищувати кілька секунд.

Після того, як тест на прогрівання було закінчено, слід перейти до перевірки результатів повного тесту на продуктивність. Для основної частини тесту на продуктивність було виконано кластер розміром 15 000 запитів у кожній архітектурі. Саме з таким розміром кластеру буде можливо наглядну різницю в ефективності роботи сервісів.

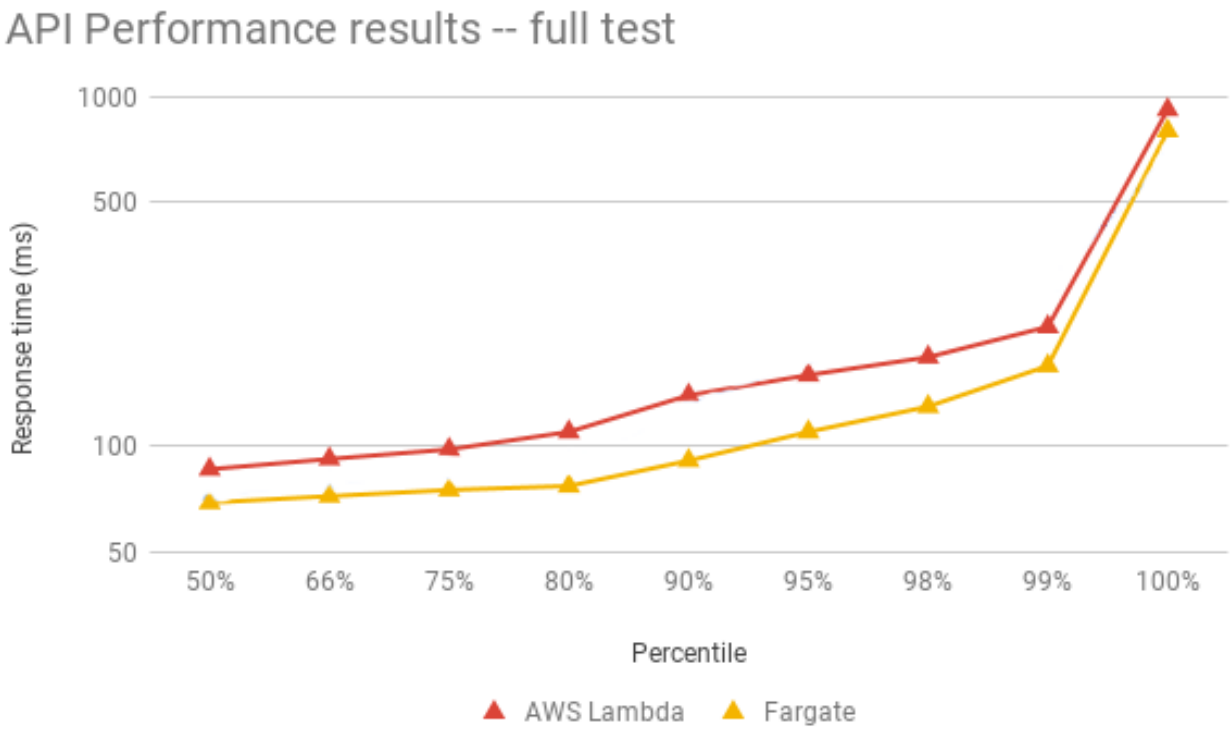

Рисунок 4.4 Графік тестування продуктивності API у режимі повного прогону

Виходячи з графіку зображеного на рисунку 4.4 можливо стверджувати що:

– AWS Fargate все ще був найшвидшим у всій плані, хоча розрив звузився. Проксі-сервіс API Gateway був майже таким же швидким, як Fargate на медіані, і AWS Lambda не відставав.

– реальні відмінності виявляються між 80-м та 99-м відсотками. Fargate мав набагато стійкіші показники, оскільки піднімався відсотків. 98-й відсотковий запит для Fargate менше, ніж удвічі більший за медіану (130 мс проти 69 мс відповідно). На відміну від цього, 98-й перцентиль для проксі-сервісу API Gateway був більш ніж втричі середнім (250 мс проти 73 мс відповідно).

– AWS Lambda перевершив проксі-сервіс API шлюзу на кілька вищих відсотків. Між 95-м та 99-м процентилами AWS Lambda насправді був швидшим, ніж проксі-сервіс API Gateway.

4.3 Висновки за проведеними дослідженнями

У випадку якщо потрібна висока продуктивність, то використання виділених інстансів із Fargate (або ECS / EKS / EC2) буде самим продуктивним рішенням. Для цього знадобиться більше налаштування та управління інфраструктурою, але це може знадобитися для вашого випадку використання.

У величезній кількості інших ситуацій використовуйте AWS Lambda. Лямбда просто розгортається (якщо ви використовуєте інструмент розгортання). Це надійний і масштабований. Вам не доведеться турбуватися про налаштування кучки ручок, щоб отримати міцну продуктивність. І це код, тому ви можете робити все, що завгодно. Я використовую його майже для всього.

ВИСНОВКИ

У результаті виконання роботи було досліджено історію технології хмарних обчислень, проведено огляд сучасних тенденцій технології у Cloud Computing сфері та виділено основні принципи надання хмарних послуг. Одним з кроків було опрацювання та вибір потрібного провайдеру хмарних обчислень. По ряду причин описаних у роботі буде обрано Amazon Web Services як основного постачальника ресурсів.

Також було поставлено проблему оптимізації хмарних обчислень з точки зору учасників бізнес-процесу, побудовано модель багаторівневої хмарної мережі та зроблено детальне порівняння сервісів хмарних обчислень за різними моделями, а саме сервісів AWS Lambda, AWS Fargate та AWS EC2.

Завдяки проробленій роботі стало можливим зосередити основні критерії порівняння сервісів, виділити операційні та фінансовий критерії. У ході роботи було виявлено Serverless напрямок як окремий для порівняння і на основі цього було створено порівняльну характеристику AWS Fargate та AWS Lambda у розробці ПЗ, фінансових питаннях та у шляхах і способах моніторингу сервісів.

Як вже зауважувалося вище, проведений аналіз навіть після реалізації усіх поставлених задач, для комерційного використання має дороблюватися. Перш за все, необхідно проведення детального аналізу продукту, який буде розроблятися з використанням хмарних технологій.

На момент написання роботи не розглядався новий сервіс AWS Outposts який може стати логічним продовженням модельної лінійки хмарних обчислень. Це дозволить провести оптимізацію програмних додатків, які потребують низької затримки доступу до локальних систем, локальної обробки даних або локального зберігання даних.

ПЕРЕЛІК ПОСИЛАНЬ

1. «Хмарні обчислення», Режим доступу: URL: https://uk.wikipedia.org/wiki/Хмарні_обчислення (Дата звернення: 25.04.2020)

2. «Хмарні обчислення», Режим доступу: URL: http://integritysys.com.ua/ solutions/pricatecloud-solution/ (Дата звернення: 25.04.2020)

3. «Дослідження у сфері Cloud», Режим доступу: URL: https://www.marketing.spb.ru/lib-mm/sales/affiliated_chain.html (Дата звернення: 25.04.2020)

4. «Навчання у хмарних технологіях», Режим доступу: URL: https://sites.google.com/site/olwarr339/ikt-u-navcanni/hmarni-tehnologiie-navcanna (Дата звернення: 25.04.2020)

5. «Gartner official statistic», Режим доступу: URL: http://www.dailycomm.ru/m/49782/ (Дата звернення: 25.04.2020)

6. «Top cloud providers 2019», Режим доступу: URL: https://www.zdnet.com/ article/top-cloud-providers-2019-aws-microsoft-azure-google-cloud-ibm-makes-hybrid-move-salesforce-dominates-saas/ (Дата звернення: 25.04.2020)

7. «AWS vs Azure vs Google Cloud: What's the best cloud platform for enterprise?», Режим доступу: URL: https://www.computerworld.com/article/3429365/aws-vs-azure-vs-google-whats-the-best-cloud-platform-for-enterprise.html (Дата звернення: 25.04.2020)

8. L. Minh Dang , Md. Jalil Piran, Dongil Han. Review A Survey on Internet of Things and Cloud Computing for Healthcare. MDPI, 2019.

9. «What is Iaas», Режим доступу: URL: https://azure. microsoft.com/en-us/overview/what-is-iaas/ (Дата звернення: 25.04.2020)

10. Kephart J., Chess D. The vision of autonomic computing // IEEE Computer Journal, 2003. — No 1. — Pp. 41-50.

11. Суханов В.И. Минимизация трафика в облачной инфраструктуре // Научный журнал КубГАУ, 2012. — № 78 (04).

12. Орлов С. Оптимізація глобальної мережі // журнал мережевих рішень, 2011. — No 9.

13. Litoiu M., Woodside M., Wong J., Ng J., Iszlai G. A business driven cloud optimization architecture // Symposium on Applied Computing, 2010. — Pp. 380-385.

14. Li J., Chineck J., Woodside M., Litoiu M. Fast Scalable Optimization to Configure Service Systems Having Cost and Quality of Service Constraints // In.: ACM/IEEE International Conference on Autonomic Computing, Barcelona, June 2009. — Pp 159-168.

15. Brun Y. Engineering Self-Adaptive System through Feedback Loops // Software Engineering for Self-Adaptive Systems., 2009. — Pp.48-70.

16. Т. Г. Бєлова, І.О. Побіженко, В.В. Побіженко. Аналіз проблем довіри в хмарних технологій // І.О. Побіженко. — Харків, 2011.

17. «What is AWS», Режим доступу: URL: https://aws.amazon.com/ru/what-is-aws/ (Дата звернення: 25.04.2020)

18. «AWS Lambda functions», Режим доступу: URL: https://aws.amazon.com/en/lambda/ (Дата звернення: 25.04.2020)

19. «AWS Fargate», Режим доступу: URL: https://aws.amazon.com/en/fargate/ (Дата звернення: 25.04.2020)

20. «AWS Fargate detailed review», Режим доступу: URL: https://www.g2.com/products/aws-fargate/reviews (Дата звернення: 25.04.2020)

21. «AWS EC2», Режим доступу: URL: https://aws.amazon.com/en/ec2/ (Дата звернення: 25.04.2020)

22. «Set Up AWS Infrastructure Monitoring Instantly», Режим доступу: URL: https://www.bluematador.com/ (Дата звернення: 25.04.2020)