УДК 004.042 https://doi.org/10.15407/pp2022.01.67

В.О.Тюрін, А.Ю. Дорошенко, О.В. Савчук

АНАЛІТИЧНЕ СХОВИЩЕ ДЛЯ ВЕЛИКИХ ПОТОКОВИХ ДАНИХ

Розроблено концепцію архітектури з організації аналітичного сховища даних на основі інфраструктури Google Cloud Platform (GCP). Проведено аналіз існуючих рішень у галузі безсерверних аналітичних сховищ. Проведено порівняльний аналіз із найбільш розповсюдженими існуючими рішеннями та здійснено експериментальне випробування розробленої концепції. Наведено рекомендації з організації сховища даних з можливістю підтримки подій із змінною схемою даних. Розроблено систему потокової передачі даних. Розроблену концепцію повністю реалізовано у GCP з метою проведення функціонального тестування.

Ключові слова: безсерверні, аналітичні сховища, BigQuery, AWS, GCP, ETL, повідомлення, потокова передача даних.

Вступ

У роботі описано концепцію архітектурного рішення щодо безсерверного (serverless) аналітичного сховища даних. Запропоновані шляхи реалізації концепції, що дають змогу створювати аналітичні сховища:

- швидкими (завдяки написанню мінімальної кількості коду);
- з доброю масштабованістю (левова частка відповідальності за масштабування передається хмарному провайдеру);
- легкими у підтримці;
- з високим рівнем захисту даних;
- з легкою інтеграцією до засобів прийняття рішень.

Наразі подібні системи мають надвисокий попит через те, що ціна сховищ стала відносно дешевою і компанії отримали змогу зберігати й використовувати для аналізу великі об'єми операційних даних [1-4]. Через те, що не усі бізнеси мають досвідчені відділи IT, багато з таких компаній зазнають невдач у процесі створення аналітичного сховища [5-6]. Створення концепції подібного сховища на базі одного з найкращих провайдерів хмарних послуг (GCP) може стати проривним як у технологічному плані, так і у плані ефективності бізнесових рішень.

Розмір фінансових інвестицій у галузь уже сягає мільярдів доларів [7-8], проте бракує саме науково-технологічних робіт та стандартизації й аналізу наявних рішень. Ця проблема існує тому, що галузь розвивається дуже швидко і ще не підхоплена університетами. Тому науково-технічна документація здебільшого створена самими розробниками систем або їх партнерами із впровадження, а кожна робота розглядає лише єдиний варіант власної реалізації. Основна мета таких робіт – це просування власного бізнесу, а не деталізований технічний аналіз запропонованих рішень.

Створений у даній роботі проєкт концепції несе науково-технічну новизну в порівнянні з існуючими рішеннями, що перевіряється на детальному аналізі й формулюванні незалежної оцінки й оформленні концепції. Ця робота також дає можливість комерційного використання, впровадження або консультації із впровадження концепції та супроводу його технічної реалізації.

У роботі розглянуті існуючі рішення та стандарти у сфері побудови хмарних аналітичних сховищ даних, здійснено їх порівняльний аналіз на основі практичних реалізацій та надана незалежна оцінка сформованій концепції.

Головна мета роботи – створення концепції аналітичних сховищ даних.

Для цього потрібно було розв'язати наступні задачі:

- розглянути існуючі рішення;
- провести аналіз;
- розглянути технічні можливості, які не були запропоновані;

• оцінити економічну складову;

• запропонувати відповідне архітектурне рішення.

Задля реалізації системи, яка відповідала б заданій меті, були проведені теоретичні та практичні дослідження у п'ять етапів:

• базова теоретична підготовка проєкту, що включала в себе теоретичну підготовку по всім архітектурним та технологічним напрямкам, які належать до сфери побудови аналітичних сховищ даних;

• теоретична підготовка в розділах хмарних систем, поглиблення знань та практичних навичок у використанні різних провайдерів хмарних послуг, вивчення сервісів, котрі надаються у використання;

• технічна реалізація концепції;

• порівняльний аналіз;

• створення технічної документації.

Проєкт є безкоштовним для користувачів, проте на основі розробки може бути створена компанія, що буде надавати послуги із впровадження моделі. Цільовий сегмент компаній, які можуть використати цей науковий проєкт, дуже широкий. Фактично це може бути будь-яка компанія, яка виросла достатньо, щоб почати використовувати свої операційні дані для прийняття рішень.

Для технічної реалізації ідей концепції були обрані сучасні хмарні технології.

Огляд платформи AWS Serverless Data Lake Framework (SDLF)

AWS Serverless Data Lake Framework (SDLF) – це стандарт (готове архітектурне рішення), засноване на компонентах (ресурсах) екосистеми AWS.

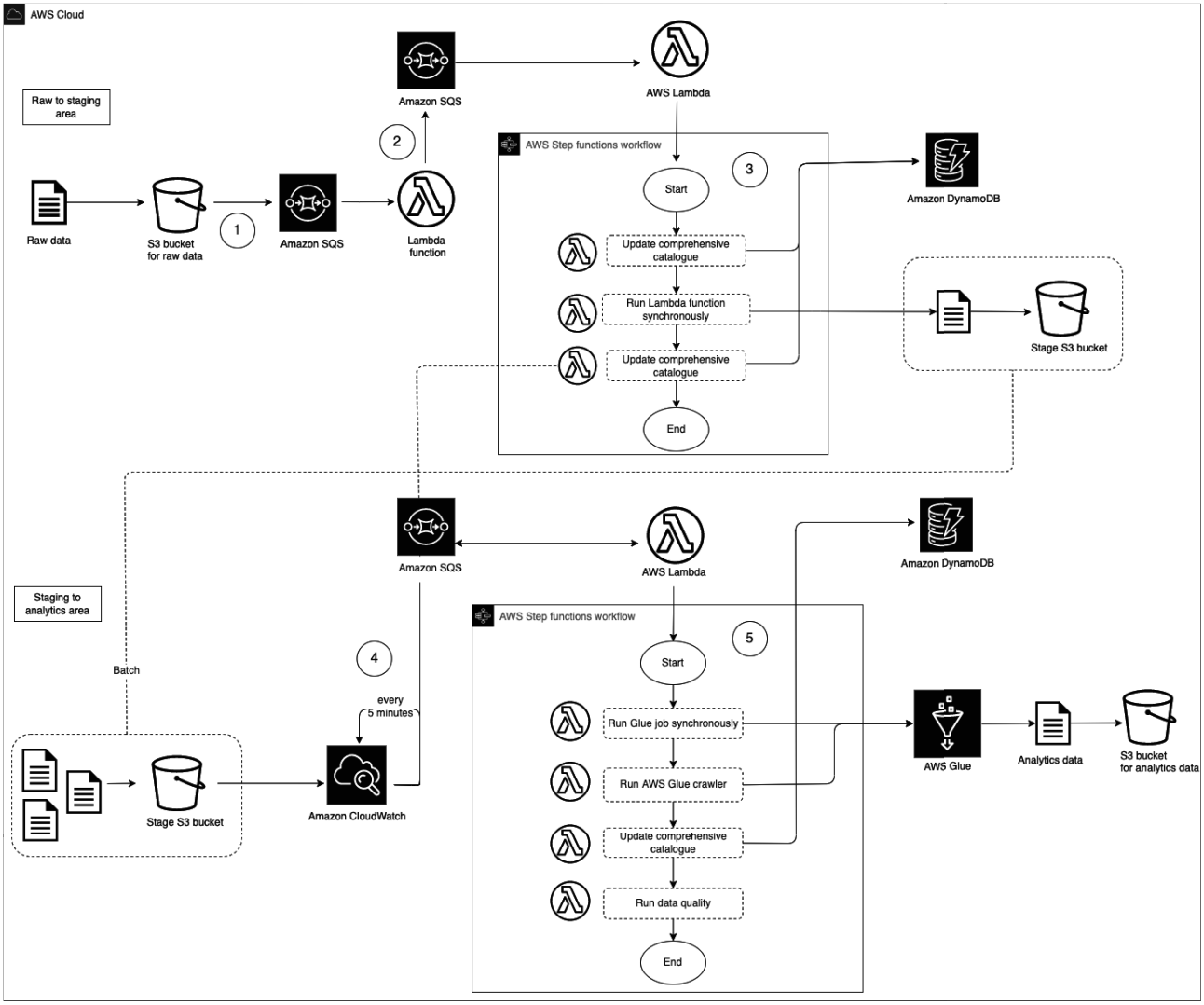

Рис. 1. Архітектурна схема AWS SDLF[10]

Це дозволяє швидко створювати «озеро даних» Data Lake, що відповідає сучасним потребам [9-10]. SDLF є головним конкурентом на ринку для розроблюваного стандарту.

Основними перевагами SDLF є:

• швидке розгортання (AWS Team надає шаблони готової інфраструктури);

• безсерверна архітектура (дозволяє значно скоротити необхідні навички і відповідно чисельність команди підтримки);

• гнучкість до змін (є можливість вставити свій код бізнес-логіки в поточну архітектуру).

До недоліків належать:

• повна залежність від AWS як провайдера інфраструктури (схема не може бути реалізована в інших хмарах);

• значні приховані затримки (latency) при високому навантаженні;

• незручний формат Landing Zone (файли із записами); у процесі реконсиляції чи під час пошуку проблеми знайти інформацію у Landing Zone майже неможливо;

• дуже складна модель контролю даних;

• невирішеність проблеми контролю схеми даних.

Використання Upsolver для IronSource DataLake

IronSource DataLake – це архітектурне рішення для побудови аналітичного сховища компанії. Компанія використала Upsolver – продукт, що полегшує створення подібних систем.

IronSource використовує Upsolver в найтяжчому для реализації місці в системі – контролі за схемами і взаємодії з продюсерами даних. Завдяки використанню ще одного продукту Kafka, керованого подіями, та регістру схем Schema Registry з'являється гарантія несуперечності схем постачальника та споживача даних.

Архітектура DataLake передбачає рівень RAW (LANDING) зберігання початкових даних – у даній архітектурі цей рівень, як і в SDLF реалізовано за допомогою Amazon S3 та файлів. Серед переваг цього підходу можна виділити дешевизну й зручність застосування одночасно декількох Query Engines до одних і тих самих даних. Проте водночас є й недоліки:

• дуже скаладна реконцеляція RAW Layer; тобто, за наявності певних проблем із несуперечністю даних перевірка даних на рівні RAW стає непростою задачею для інженерів;

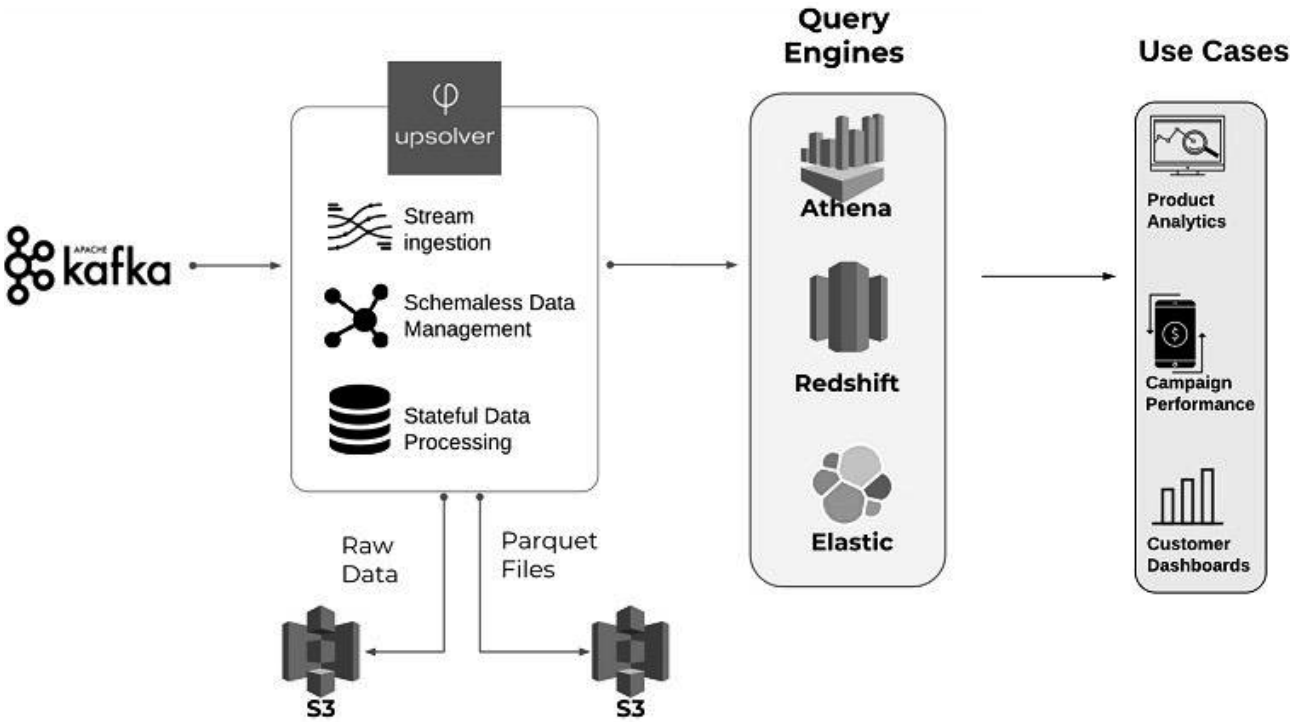

Рис. 2. Архітектурна схема IronSource DataLake[11]

• необхідність додаткових інвестицій у моніторінг системи для створення засобів контролю за збереженням несуперечності;

• подвійна вендорна залежність – компанія залежить і від AWS, як від провайдера хмарних сервісів, і від Upsolver як провайдера платформи інтеграції із постачальниками даних.

SimilarWeb DataLake

SimilarWeb DataLake – це архітектурне рішення компанії для побудови системи Anomaly Detection на Web сторінках.

Виходячи із особливостей використання системи очевидно, що подібне озеро DataLake орієнтоване на короткотривале збереження початкових даних через їх величезну кількість. Система має адаптуватися до неструктурованих даних через те, що компанії необхідно фільтрувати значну кількість веб-сторінок, розроблених різними компаніями, які, відповідно, не стандартизовані.

SimilarWeb також використовує Upsolver для вирішення проблеми потокової обробки потоків даних. На схемі можна побачити, що початкові дані зберігаються в окремому сховищі S3 лише 1 день (термін, необхідний для обробки даних), далі події групуються у логічні групи, виділяються унікальні ключі тощо.

Зі схеми можна також побачити, що SimilarWeb виеористовує AWS Glue Data Catalog для контролю схеми даних. Компанія будує супер-сет моделі даних (конкетенація можливих полів) для подальшого використання у запитах Athena. Athena використовується як система побудови запитів до даних і виявлення даних, що відрізняються від заданих правил або правил розподілу.

Подібна система має обмеження і не може бути повністю автоматизованою, бо різні потоки подій можуть конфліктувати між собою, тому їх потрібно розділяти.

Різні дані одного й того ж потоку також можуть конфліктувати один з одним, тому для реалізації потокової обробки компанія і вирішила використати готовий продукт.

Таким чином, серед значних недоліків цієї архітектури можна також зазначити:

• подвійну залежність від провайдерів послуг (AWS для ресурсів та Upsolver, що повністю контролює логіку потокової обробки);

• відсутність Data Marts і можливості повторно перевикористати вже існуючі обчислення;

• обмеження Athena як провайдера машини SQL.

Опис запропонованої концепції

Запропонована схема має наступний алгоритм роботи:

Потокова передача даних у Pub/Sub – вхідну точку для даних до хмари.

Потокова передача даних із Pub/Sub до Landing Zone.

Паралельний запис кожної події в контролюючу таблицю.

Обраховування зміни даних (дельти) за обраний часовий проміжок.

Оновлення даних по виборці дельти.

Побудова Data Marts.

Підготовка даних для використання кінцевими сервісами-користувачами (ма-

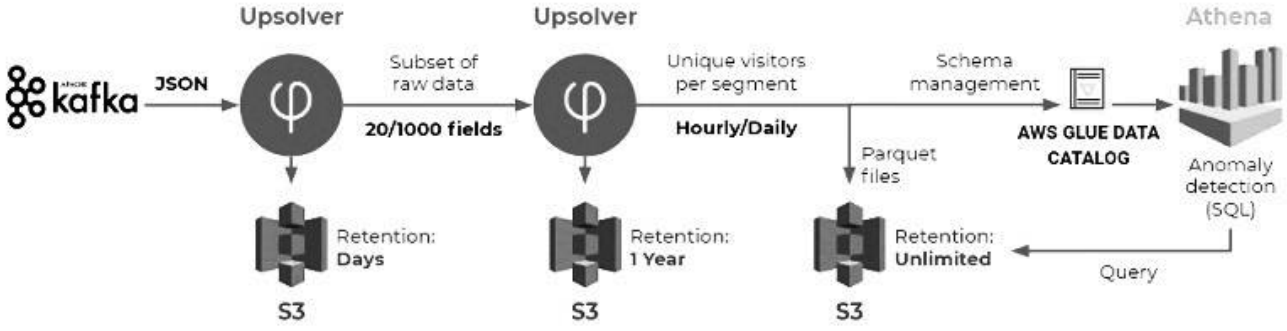

Рис.3. Архітектурна схема SimilarWeb DataLake[12]

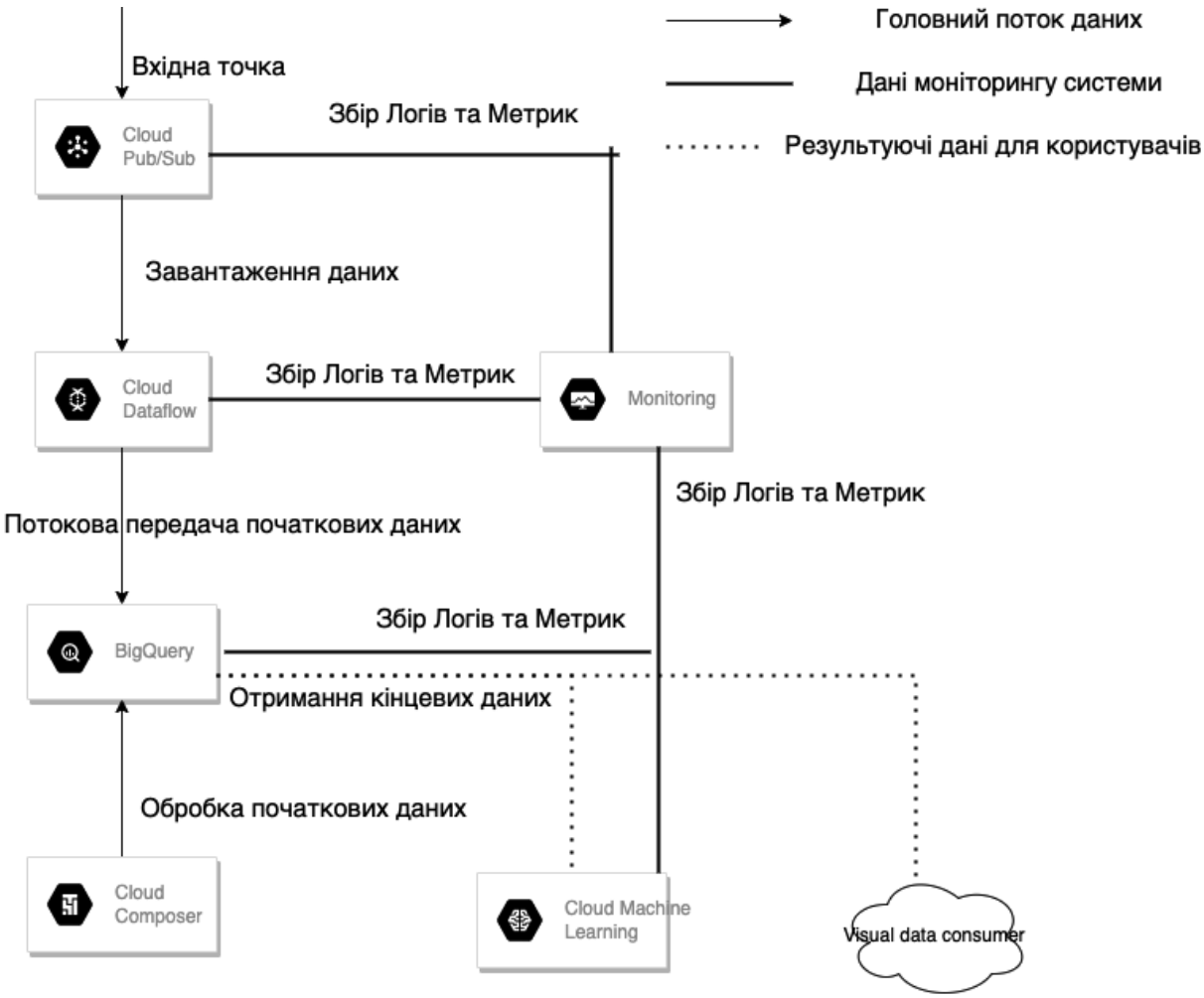

Рис. 4 Запропонована архітектурна схема із використанням GCP

шинне навчання (МЛ), засоби візуалізації, засоби прийняття рішень);

Паралельний моніторинг процесів і ключових метрик.

Засоби безпечного доступу до даних

На сьогодні існує багато загроз, що можуть призвести до несанкціонованого доступу до даних. Такі загрози можуть бути зовнішніми, як - от:

- хакерська атака;
- кібер-атака іншої держави або терористів;
- атака зі сторони іншої компанії або третіх сторін.

Також загрози можуть бути і внутрішніми від джерел (або людей), що навіть не знають про це:

- шахрайство (fraud);
- атака колишнього/діючого працівника з певним рівнем доступу;
- бездіяльність або приховування (працівник міг зробити помилку, що призвела до витоку даних і під страхом наслідків він може приховати факт події);
- ненавмисне заволодіння даними під час розробки чи підтримки системи.

Багато реалізацій аналітичних сховищ мають проблеми із забезпеченням безпеки, особливо в останньому пункті зі списку. Через низький рівень автоматизації такі системи вимагають постійного доступу до даних користувачів для розробників системи.

Запропонована концепція не має такого недоліку, надання доступу до даних dataset відбувається за допомогою Identity and Access Management(IAM), тому лише авторизовані для цієї операції можуть отримати доступ.

Якщо лише частина даних є таємницею, наприклад, userId – не є секретом, бо не є PII, а email користувача – має бути захищеним, то в GCP існує ресурс Data Catalog, який передбачено використовувати під час реалізації концепції для того,

щоб захистити окрему колонку даних. Data Catalog дозволяє створити ярлик (label) на колонку із даними, додати його до групи і надати доступ до групи лише авторизованим користувачам. Тож концепція значно скорочує і визначає коло осіб, що мають доступ до даних користувачів системи.

Для захисту від зовнішніх загроз існує практика не надання прямого доступу до аналітичного сховища стороннім сервісам, навіть тієї ж самої компанії.

Запропонована концепція реалізує цей засіб захисту – доступ до сховища неможливий із зовнішнього середовища, точкою взаємодії із зовнішнім світом є Pub/Sub – для завантаження даних, що передаються потоком Dataflow, який написано інженерами, що розуміють будову сховища і несуть за нього відповідальність. Доступ до Data Mart для користувачів є «тільки для читання», тобто вони не можуть змінити дані, а весь доступ також є авторизованим (і для людей, і для сервісів за допомогою Service Account).

Порівняльний аналіз моделей

Запропонована концепція має значну перевагу у надійності Entry Point. Якщо в SDLF – це сервіс, що зберігає файли (S3), а файли з даними можуть бути великими, то цей стандарт може мати проблеми із незавершеним завантаженням і відповідно втратою даних. На відміну від цього запропонована концепція використовує Pub/Sub [13-16], що здійснює принцип "at least once delivery", який гарантує доставку повідомлення один раз або більше, що виключає можливість втрати даних на цьому кроці. Upsolver використовує для цих цілей Kafka Schema Registry та дає своїм користувачам можливість контролю схеми даних. SimilarWeb додатково використовує AWS Glue Schema Registry для того, щоб мати схему ще й на рівні машини запитів Query Engine, якою в даному випадку є AWS Athena. Kafka є еквівалентною до запропонованого рішення за надійністю, проте значно складнішою у налаштуванні та підтримці (навіть у безсерверній реалізації).

Запропонована концепція має значну перевагу у методі потокової передачі даних над головним конкурентом – SDLF. Він використовує технологію Dataflow [17-18], що працює на шаблонах, які реалізують Apache Beam – провідний стандарт у сфері сховищ даних ETL [17]. Це дозволяє використовувати здобутки бібліотек із відкритим кодом і писати складні потоки із використанням малої кількості самописного коду. Крім того, Dataflow запускається на контрольованому кластері з машин, на відміну від AWS Lambda [19] в SDLF. Це означає, що система краще масштабується і коштує значно дешевше навіть за умови великих і дуже великих даних. Як видно із досвіду IronSource та SimilarWeb багато компаній не бажають мати справу з потоковою обробкою великих даних й купують вже готові сервіси. Однак інженери компанії все одно мають розуміти, як працює їх потокова обробка для того, щоб знати, чи задовольняє реалізація вендора їх бізнес-потреби. Додатково це значно збільшує залежність від провайдера послуг. Запропонована концепція має значну перевагу над архітектурами IronSource та SimilarWeb, тому що використовує бібліоеки з відкритим кодом.

Запропонована концепція має значну перевагу в організації зони Landing над усіма конкурентами за рахунок Dataset в BigQuery. Це дозволяє робити запити до даних мовою SQL, переглядати їх, конвертувати, проводити реконсиляцію (перевірку несуперечності системи) на відміну від інших, де дані фізично зберігаються у файлах, тож пошук конкретного запису фактично неможливий. Серед недоліків цього підходу можна зазначити вищий рівень витрат на сховище, проте якщо брати повну вартість володіння систем TCO, а не окремі витрати на сховище, то різниця буде не такою великою, бо підтримка стає простішою (означає, що треба витрачати менше людино-годин кваліфікованих працівників), немає потреби у окремій системі моніторингу за несуперечністю, результати обробки даних можна перевикористовувати.

Запропонована концепція має значну перевагу у моделі оновлення даних,

адже ця операція виконується за допомогою відкритої бібліотеки Airflow [21], створеної для керування складними ETL-процесами і вже реалізовує дуже багато логіки всередині, яка може бути перевикористана на відміну від реалізації SDLF, де процес обробки даних передано на AWS Lambda, тобто на самописний код.

Ще однією перевагою Airflow є використання кластеру виконувачів на основі Kubernetes. Тобто оплата відбувається за кількістю використаних годин і потужністю інфраструктури кластеру, а не за кількістю викликів. Таким чином запропонована модель обробки даних буде значно дешевшою на великих даних, ніж SDLF.

Вразливість моделей до зміни схеми даних

Однією з найбільших проблем аналітичних баз даних є необхідність адаптуватися до зміни початкових даних, водночас зберігаючи якість кінцевих даних. Очевидним недоліком SDLF є повна відсутність контролю схеми даних, що унеможливлює побудову повноцінного сховища.

Якщо компанія бере на використання SDLF, то підтримка схеми лягає на відповідальність сервісів озера даних Data Lake. Такий підхід до архітектури може бути правильним, проте ця особливість не підкреслена в документації стандарту для того, щоб мати змогу говорити про "дуже швидке розгортання SDLF від розробки до Production". Однак необхідність самостійної інтеграції контролю схем, наприклад, за допомогою Kafka Schema Registry[13-15], забере багато додаткового часу.

На відміну від SDLF, запропонована концепція може працювати у двох режимах.

Перший режим аналогічний до SDLF. Він передбачає передачу відповідальності за схему даних попередньому сервісу, тобто архітектура передбачає, що до сховища доходять лише валідні повідомлення.

Другий режим передбачає використання Pub/Sub Schema Registry, що буде перевіряти схему повідомлення на етапі спроби відправки сервісом постачальника даних. Постачальники, які не будуть відправляти повідомлення у схемі, підтримуваної сховищем, просто не зможуть відправити таке повідомлення й отримають зрозумілу відповідь про помилку.

Такий підхід має значну перевагу через те, що помилки виявляються на першому ж кроці, коли постачальник намагається відправити повідомлення, а не на кінцевому, як в SDLF, коли дані вже у сховищі, проте їх неможливо обробити, через те, що схема змінилася або зовсім невалідна.

Важливо зазначити, що у великих системах, де кількість унікальних типів подій може вимірюватись сотнями і тисячами, в обох концепціях (запропонованому та SDLF стандарті), слід делегувати відповідальність за валідацію схем централізованому сховищу Event Store.

Висновки

У роботі розроблено концепцію організації аналітичного сховища даних, а саме:

- метод взаємодії постачальників даних зі сховищем;
- метод контролю схеми даних;
- метод потокової передачі даних;
- метод зберігання початкових даних;
- метод обробки даних;
- метод надання безпечного доступу до даних.

Розглянуто інші існуючі на ринку стандарти, а саме:

- SDLF – провідний стандарт рекомендований AWS;
- IronSource DL із використанням Upsolver;
- SimilarWeb DL із використанням Upsolver.

Проведено порівняльний аналіз (здебільшого з SDLF, позаяк його реалізація відкрита, а деталі реалізацій приватних компаній є таємницею). Детально оглянуто переваги запропонованої концепції над існуючими. Наведено рекомендації щодо способу інтеграції концепції із застосунками із контролю схеми даних.

Розроблено сервіс потокової передачі даних за допомогою Apache Beam мовою Java.

References

1. *Електронний ресурс [01.02.2022]: https://aws.amazon.com/data-warehouse/*
2. *Електроний ресурс [01.02.2022]: https://cloud.mts.ru/cloud-thinking/blog/data-warehouse/*
3. Електроний ресурс [01.02.2022]: https://azure.microsoft.com/en-us/resources/customer-stories
4. Електроний ресурс [01.02.2022]: https://builtin.com/data-science/company-data-lake-questions
5. Електроний ресурс [01.02.2022]: https://databricks.com/discover/data-lakes/challenges
6. Електроний ресурс [01.02.2022]: https://lingarogroup.com/blog/8-challenges-faced-by-ctos-when-starting-data-lake-projects/
7. *Електроний ресурс [01.*02.2022]: https://www.qlik.com/blog/2020-the-year-cloud-data-warehouses-arrived
8. Електроний ресурс [01.02.2022]: https://www.castordoc.com/blog/cloud-data-warehousing-the-past-present-and-future
9. *Електроний ресурс [01.02.2022]: https://catalog.us-east-1.prod.workshops.aws/v2/workshops/501cb14c-91b3-455c-a2a9-d0a21ce68114/en-US*
10. *Електроний ресурс [01.02.2022]: https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/deploy-and-manage-a-serverless-data-lake-on-the-aws-cloud-by-using-infrastructure-as-code.html*
11. Електроний ресурс [01.02.2022]: https://www.upsolver.com/case-studies/ironsource-how-built-petabyte-scale-data-lake
12. Електроний ресурс [01.02.2022]: https://aws.amazon.com/blogs/big-data/how-similarweb-analyze-hundreds-of-terabytes-of-data-every-month-with-amazon-athena-and-upsolver/
13. *Електроний ресурс [01.02.2022]: https://docs.confluent.io/platform/current/schema-registry/index.html*
14. *Електорний ресурс [01.02.2022]: https://habr.com/ru/company/alfastrah/blog/547092/*
15. *Електроний ресурс [01.*02.2022]: https://www.bigdataschool.ru/blog/kafka-big-data-schema-registry.html
16. *Електроний ресурс [01.*02.2022]: https://cloud.google.com/pubsub/docs/schemas
17. Електроний ресурс [01.02.2022]: https://cloud.google.com/dataflow
18. Електроний ресурс [01.02.2022]: https://habr.com/ru/post/122479/
19. Електроний ресурс [01.02.2022]: https://beam.apache.org/
20. Електроний ресурс [01.02.2022]: https://aws.amazon.com/ru/lambda/
21. Електроний ресурс [01.02.2022]: https://airflow.apache.org/

Отримано: 18.02.2022

***Про авторів*:**

Тюрін Валерій Олександрович,
магістрант Національного
Технічного Університету України
«КПІ імені Ігоря Сікорського».

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу теорії
комп'ютерних обчислень, професор
кафедри інформаційних систем
та технологій Національного
Технічного Університету України
«КПІ імені Ігоря Сікорського».
Кількість наукових публікацій
в українських виданнях – понад 190.
Кількість наукових публікацій
в зарубіжних виданнях – понад 80.
Індекс Хірша – 6.
http://orcid.org/0000-0002-8435-1451

Савчук Олена Володимирівна,
кандидат технічних наук, доцент кафедри
інформаційних систем та технологій
Національного Технічного Університету
України «КПІ імені Ігоря Сікорського».
Кількість наукових публікацій
в українських виданнях – понад 100.
Кількість наукових публікацій
в зарубіжних виданнях – 3.
Індекс Хірша – 2.
http://orcid.org/ 0000-0003-3176-7952

Місце роботи авторів:

Національний технічний університет
України «Київський політехнічний
інститут імені Ігоря Сікорського»,
проспект Перемоги 37
та Інститут програмних систем
НАН України, 03187, м. Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 3559
E-mail: tiurinvalery@gmail.com
doroshenkoanatoliy2@gmail.com
elenasavchuk0@gmail.com