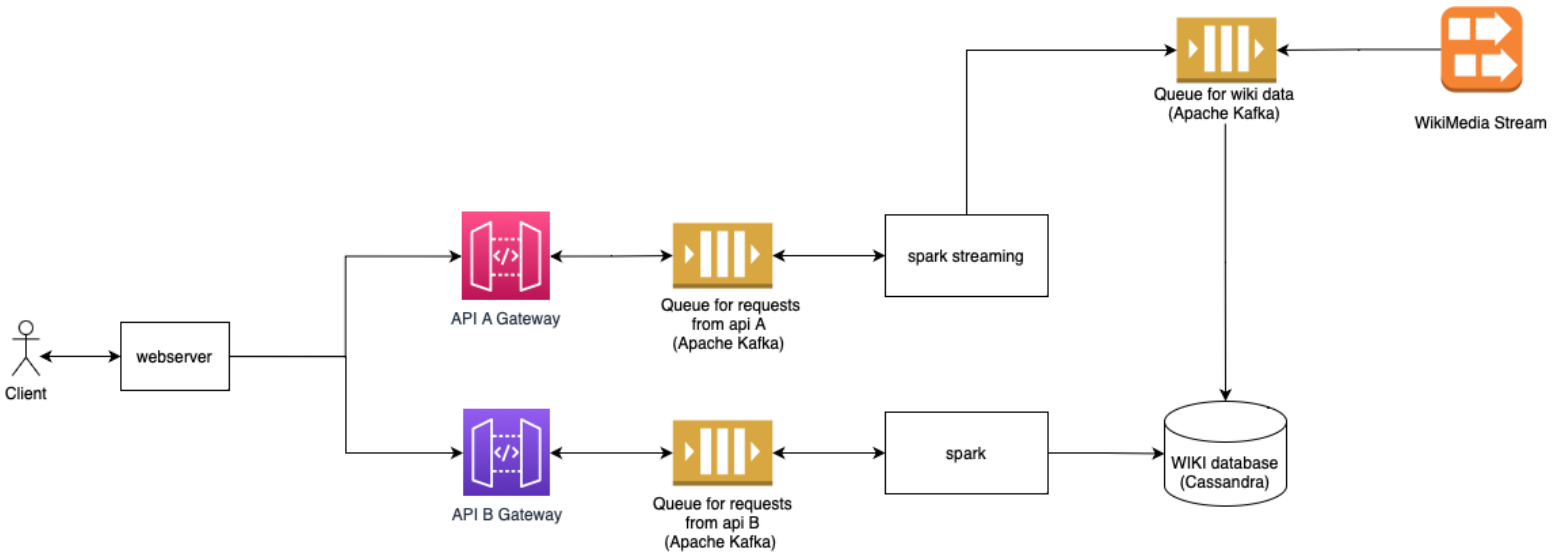
Wikimedia System Document

Detailed design documentation



Клієнт — користувач, який може взаємодіяти із системою.

Веб-сервер — єдиний спосіб взаємодії із системою через локальний порт браузера.

API A Gateway — веб сервер, який координує взаємодію клієнта з платформою. Усі запити, які надходять від клієнта і стосуються API A, йдуть через цей сервер. Реалізовано через REST API.

API B Gateway — веб сервер, який координує взаємодію клієнта з платформою. Усі запити, які надходять від клієнта і стосуються API B, йдуть через цей сервер. Реалізовано через REST API.

Є саме два "гейтвеї", оскільки вони реалізують окрему логіку над обробкою даних. Фактично, це не гейтвеї, а більше два окремих сервери, які обробляють окремі запити від клієнта і далі звертаються до різних компонент, які реалізовують логіку, потрібну для видачі результату саме для цього API.

Черга повідомлень для API A — для зменшення навантаження на систему та забезпечення надійності, усі запити, що стосуються API A, надсилаються у чергу повідомлень. При цьому в повідомленні вказується метод обробки даних для отримання результату, на який є запит, і унікальне айді запиту. Черга має два

топіки: request-topic і response-topic. Запити надсилаються у перший топік, результат читається із другого топіку. Реалізовано за допомогою Kafka.

Черга повідомлень для API B — аналогічно до попередньої черги, включно із функціоналом, топіками та інструментами для реалізації. Дві черги створено для ізоляції функціоналу одного API від іншого.

У цьому випадку, як варіант, можна було б створити, наприклад, 4 топіки (request-topic-A, response-topic-A, request-topic-B, response-topic-B), адже маємо усього два окремих API. Але в загальному, якби потрібно було додати нові ендпоінти чи реалізувати повністю нову логіку, створивши нові сервіси для API, наявність однієї черги для всіх уповільнила б роботу сервісів та викликала б непорозуміння в плані логічної ізоляції окремих компонент одна від одної.

Сервіс для стрімінгової обробки даних — реалізовано на основі Spark Streaming. Сервіс є основним “мозком” системи відносно API A, адже виконує всі обрахунки та операції над даними. Оскільки логіка API A така, що потрібно аналізувати дані за останні n годин, то вигідно брати дані як потік за потрібний час, а не піднімати всю датабазу для обрахунків. Тому цей сервер взаємодіє із двома чергами повідомлень: 1) для взаємодії із API A Gateway, читаючи запити та надсилаючи результат обробки даних; 2) для читання даних, які стрімом надходять із веб-серверу Wikimedia. Таким чином, ми забезпечуємо те, що будуть братися найбільш актуальні дані і швидкість обчислень буде більшою за рахунок того, що не будуть виконуватися запити безпосередньо до цілої датабази.

Черга для даних із Wikimedia — реалізовано на основі Kafka. Оскільки сайт оновлюється постійно, і дані надходять стрімом, ми не можемо записати все за один раз в датабазу. (Теоретично, можемо, але дані будуть доступні тільки до моменту, коли здійснений запис. Для отримання новіших даних потрібно буде знову звертатися на вебсайт і оновлювати дані. Можна було б реалізувати запис через кеш (in-memory БД Redis, наприклад), звертатися до вебсторінки раз на n секунд/хвилин/тощо, і перевіряти, чи не оновилися дані. Але обсяг даних на сторінці дуже великий, і кешувати їх всі — дуже неефективно і затратно в плані пам’яті). Тому дані потоком надходять у чергу повідомлень і поступово записуються в базу даних.

Сховище wiki даних — база даних на основі інструменту Cassandra. Оскільки об’єми даних великі, вони не структуровані та вимагають швидкого доступу до них, вибрано саме цей інструмент. У нашій системі немає вимог до consistency чи availability даних (мається на увазі, відсутній чіткий вибір між AP і CP), тому додаткових налаштувань Cassandra не має.

Сервіс для батч обробки даних — реалізовано на основі Spark. Сервіс є основним “мозком” системи відносно API B, адже виконує всі обрахунки та операції над даними (аналогічно до Сервісу стрімінгової обробки даних). Оскільки логіка API B така, що потрібно виконати аналіз даних за весь час, то використовуємо тут саме батч обробку даних, доступаючись до них безпосередньо через Cassandra. Сервіс взаємодіє із чергою повідомлень для комунікації із API B Gateway, читаючи запити та надсилаючи необхідний результат.

WikiMedia stream — сторонній вебсайт (за посиланням), на якому відображаються дані у потоці, які нам треба обробити і використати.

Data models description

Дані медіа-потоку Wiki є великими та можуть включати велику кількість оновлень у реальному часі, редагувань та взаємодії з користувачем. Оскільки Cassandra має розподілену архітектуру, вона дозволяє масштабувати її горизонтально, додаючи більше вузлів до кластера. У результаті ми можемо працювати з високою пропускною здатністю запису та читання, що робить його придатним для зберігання та обслуговування швидкоплинних медіа-потоків.

З огляду на те, які ендпоінти потрібно реалізувати в API обрані наступні таблиці з даними для збереження в сховищі з тих, що передаються медіа потоком Wiki:

1. Таблиця з композитним ключем з `page_id` та `domain`
   Такий ключ потрібен для того, щоб мати змогу повернути список існуючих доменів, для яких були створені сторінки та кількість сторінок, створених для вказаного домену, оскільки в першому випадку звертатимуться по `page_id` та в другому по `domain`

```
(uid text,
 domain text,
 rev_timestamp text,
 user_is_bot boolean,
 user_id text,
 PRIMARY KEY ((uid), domain))
```

   `uid` – унікальний ключ для даних який береться з `page_id`
   `domain` – домен Вікіпедії для даної сторінки
   `rev_timestamp` – часова позначка для даної сторінки
   `user_is_bot` – True, якщо користувач є роботом та False, якщо реальною людиною

`user_id` – id користувача

```
cqlsh:wiki> SELECT * FROM pages_by_domain;
```

uid	domain	rev_timestamp	user_id	user_is_bot
3953215	id.wikipedia.org	2023-06-08T08:09:58Z	1415671	True
4787929	ja.wikipedia.org	2023-06-08T08:10:07Z	1906552	True
9101936	mg.wiktionary.org	2023-06-08T08:10:08Z	380	True
73985218	en.wikipedia.org	2023-06-08T08:10:07Z	7701499	True
697993	sk.wikipedia.org	2023-06-08T08:10:02Z	236075	True
132785806	commons.wikimedia.org	2023-06-08T08:09:57Z	10101913	True
132785808	commons.wikimedia.org	2023-06-08T08:10:04Z	8873	True
113703004	www.wikidata.org	2023-06-08T08:10:04Z	588882	True
9101934	mg.wiktionary.org	2023-06-08T08:10:02Z	380	True
9101935	mg.wiktionary.org	2023-06-08T08:10:03Z	380	True
113703003	www.wikidata.org	2023-06-08T08:10:03Z	8461	True
1323171	mk.wikipedia.org	2023-06-08T08:10:08Z	31127	True
132785807	commons.wikimedia.org	2023-06-08T08:10:03Z	5142080	True

2. Таблиця з композитним ключем з `page_id` та `user_id`
   Такий ключ потрібен для того, щоб мати змогу повертати всі сторінки, створені користувачем із вказаним `user_id` та сторінку з указаним `page_id`

```
(uid text,
 domain text,
 rev_timestamp text,
 user_is_bot boolean,
 user_id text,
 PRIMARY KEY ((uid), user_id))
```

   Опис колонок аналогічний

3. Таблиця з композитним ключем з `page_id` та `rev_timestamp`
   Такий ключ потрібен для того, щоб мати змогу повертати ідентифікатор, назву та кількість створених сторінок усіх користувачів, які створили принаймні одну сторінку за вказаний проміжок часу (власне `rev_timestamp`).

```
(uid text,
 page_title text,
 rev_timestamp text,
 user_id text,
 PRIMARY KEY ((uid), rev_timestamp))
```

   `page_title` – назва сторінки
   Решта колонок аналогічні