ТИНЬКОФФ

Thread Wars: войны клонов на страницах Wiki

09.10.2023

Предыстория

Факультет Математики и
Компьютерных Наук
СПбГУ

Сириус

Александр Шахов

- Java-разработчик в Тинькофф.Город
- Преподаватель в СПбГУ, ЛЭТИ, Центральном университете
- Создатель книжного клуба .rar

Даниил Любаев

- Магистр МКН СПбГУ
- Преподаватель на МКН
- Разработчик на Go
- Любитель concurrency

Что такое WikiGame

- Даются две статьи на Википедии

Что такое WikiGame

- Даются две статьи на Википедии
- Нужно кликать по ссылкам и найти путь от первой до второй

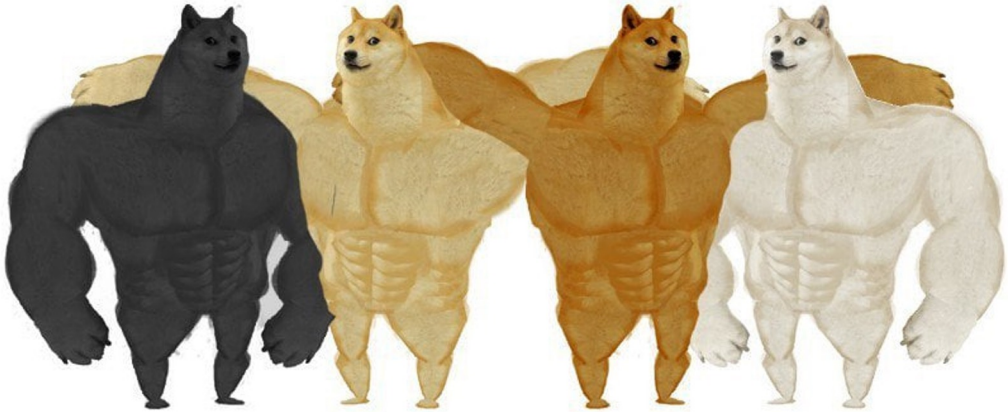
Что такое WikiGame

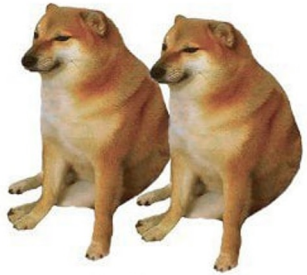
- Даются две статьи на Википедии
- Нужно кликать по ссылкам и найти путь от первой до второй
- Кто быстрее – тот и победил

Как все происходило

- Сделали демку на асинхронном Python
- Дали школьникам с ней играться
- Параллельно объясняли, что к чему
- Думали, сможет ли кто-то сделать решение, близкое по времени к нашему
- Но…

Школьники





Мы


Давай. Вошли и вышли, приключение
на 20 минут.

Постановка задачи

- Википедия умеет отдавать все ссылки на странице

Постановка задачи

```
public interface WikiDataSource {
    List<String> getLinksByTitle(String title);
}
```

Постановка задачи

```
public interface WikiDataSource {
    List<String> getLinksByTitle(String title);
}

public interface WikiRepository {
    List<String> getLinksByTitle(String title);
}
```

Постановка задачи

```
public interface WikiDataSource {
    List<String> getLinksByTitle(String title);
}

public interface WikiRepository {
    List<String> getLinksByTitle(String title);
}

public interface WikiGame {
    List<String> play(String startPageTitle, String endPageTitle, int maxDepth);
}
```

Реализации

```
@Override
public List<String> getLinksByTitle(String title) {
    try {
        String responseBody = httpClient.send( /* Create HTTP Request */).body();

        WikiLinksResponse response = /* Deserialize from responseBody */;

        return parseResponse(response);
    } catch (Throwable e) {
        throw new RuntimeException(e);
    }
}
```

Реализации

```
public class WikiRepositoryImpl implements WikiRepository {

    private final WikiDataSource wikiDataSource;

    public WikiRepositoryImpl(WikiDataSource wikiDataSource) {
        this.wikiDataSource = wikiDataSource;
    }

    @Override
    public List<String> getLinksByTitle(String title) {
        return wikiDataSource.getLinksByTitle(title);
    }
}
```

Baseline: Синхронное решение

- Простой обход в ширину (BFS – Breadth-First Search) от начала страницы
- Останавливаемся, когда встречаем страницу с нужным названием

```java
@Override
public List<String> play(String startPageTitle, String endPageTitle, int maxDepth) {
    var startedPage = new Page(startPageTitle, null);
    Queue<Page> rawPages = new LinkedList<>(Collections.singleton(startedPage));
    Queue<Page> parsedPages = new LinkedList<>();
    Set<String> parsedTitle = new HashSet<>();
    var parentEndPage = startedPage;
    do {
        var curPage = rawPages.poll();
        parentEndPage = curPage;
        var newLinks = wikiDataSource.getLinksByTitle(curPage.getTitle());
        parsedPages.add(curPage);
        parsedTitle.addAll(newLinks);
        rawPages.addAll(newLinks.stream().map(m -> new Page(m, curPage)).toList());
    } while (!parsedTitle.contains(endPageTitle));
    parsedPages.add(new Page(endPageTitle, parentEndPage));

    return getResultPath(parsedPages, endPageTitle);
}
```

```java
@Override
public List<String> play(String startPageTitle, String endPageTitle, int maxDepth) {
    var startedPage = new Page(startPageTitle, null);
    Queue<Page> rawPages = new LinkedList<>(Collections.singleton(startedPage));
    Queue<Page> parsedPages = new LinkedList<>();
    Set<String> parsedTitle = new HashSet<>();
    var parentEndPage = startedPage;
    do {
        var curPage = rawPages.poll();
        parentEndPage = curPage;
        var newLinks = wikiDataSource.getLinksByTitle(curPage.getTitle());
        parsedPages.add(curPage);
        parsedTitle.addAll(newLinks);
        rawPages.addAll(newLinks.stream().map(m -> new Page(m, curPage)).toList());
    } while (!parsedTitle.contains(endPageTitle));
    parsedPages.add(new Page(endPageTitle, parentEndPage));

    return getResultPath(parsedPages, endPageTitle);
}
```

```java
@Override
public List<String> play(String startPageTitle, String endPageTitle, int maxDepth) {
    var startedPage = new Page(startPageTitle, null);
    Queue<Page> rawPages = new LinkedList<>(Collections.singleton(startedPage));
    Queue<Page> parsedPages = new LinkedList<>();
    Set<String> parsedTitle = new HashSet<>();
    var parentEndPage = startedPage;
    do {
        var curPage = rawPages.poll();
        parentEndPage = curPage;
        var newLinks = wikiDataSource.getLinksByTitle(curPage.getTitle());
        parsedPages.add(curPage);
        parsedTitle.addAll(newLinks);
        rawPages.addAll(newLinks.stream().map(m -> new Page(m, curPage)).toList());
    } while (!parsedTitle.contains(endPageTitle));
    parsedPages.add(new Page(endPageTitle, parentEndPage));

    return getResultPath(parsedPages, endPageTitle);
}
```

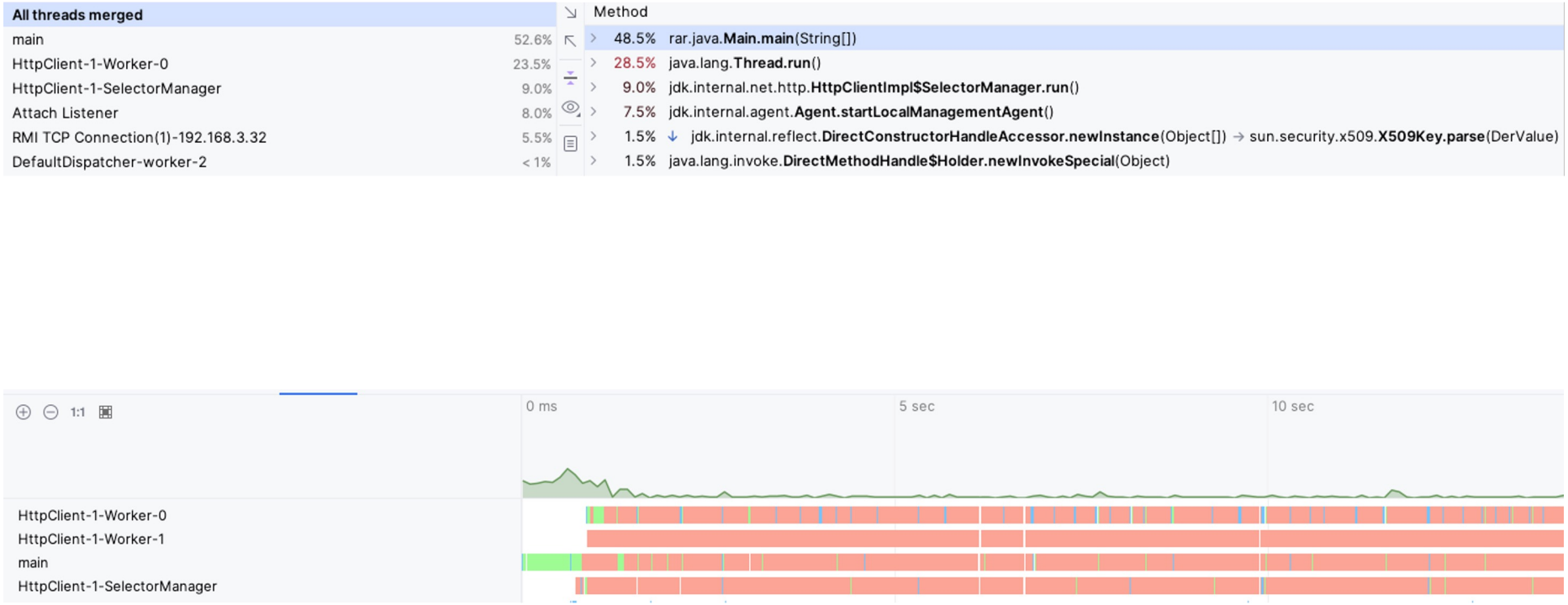
```java
@Override
public List<String> play(String startPageTitle, String endPageTitle, int maxDepth) {
    var startedPage = new Page(startPageTitle, null);
    Queue<Page> rawPages = new LinkedList<>(Collections.singleton(startedPage));
    Queue<Page> parsedPages = new LinkedList<>();
    Set<String> parsedTitle = new HashSet<>();
    var parentEndPage = startedPage;
    do {
        var curPage = rawPages.poll();
        parentEndPage = curPage;
        var newLinks = wikiDataSource.getLinksByTitle(curPage.getTitle());
        parsedPages.add(curPage);
        parsedTitle.addAll(newLinks);
        rawPages.addAll(newLinks.stream().map(m -> new Page(m, curPage)).toList());
    } while (!parsedTitle.contains(endPageTitle));
    parsedPages.add(new Page(endPageTitle, parentEndPage));

    return getResultPath(parsedPages, endPageTitle);
}
```

Замеры: Синхронное решение

Начальная страница	Конечная страница	Школьники	Синхронное решение
Бакуган	Библия	0,73 сек.	> 30 мин.
!!!	Теория гомологий	0,87 сек.	…
А (кириллица)	Йод	0,84 сек.	…
!!!!!!!	Физика	0,76 сек.	…
Бутерброд	Бродский	0,53 сек.	...
Охотники за привидениями	Пуджа	0,75 сек.	…

Синхронное решение – долго

- Должны ожидать каждый запрос


All threads merged
main 52.6%
HttpClient-1-Worker-0 23.5%
HttpClient-1-SelectorManager 9.0%
Attach Listener 8.0%
RMI TCP Connection(1)-192.168.3.32 5.5%
DefaultDispatcher-worker-2 < 1%
Method
48.5% rar.java.Main.main(String[])
28.5% java.lang.Thread.run()
9.0% jdk.internal.net.http.HttpClientImpl$SelectorManager.run()
7.5% jdk.internal.agent.Agent.startLocalManagementAgent()
1.5% jdk.internal.reflect.DirectConstructorHandleAccessor.newInstance(Object[]) → sun.security.x509.X509Key.parse(DerValue)
1.5% java.lang.invoke.DirectMethodHandle$Holder.newInvokeSpecial(Object)
1:1
0 ms
5 sec
10 sec
HttpClient-1-Worker-0
HttpClient-1-Worker-1
main
HttpClient-1-SelectorManager

Что делать?

- Каждая ветка страниц в дереве BFS *почти независима* друг от друга
- Нужно лишь пропускать страницы, на которых мы уже были
- Остальное можем обрабатывать параллельно

Executors

```java
@Override
public List<String> play(@NotNull String startPageTitle, @NotNull String endPageTitle, int maxDepth) {
   var startPage = new Page(startPageTitle, null);
   Queue<Page> rawPages = new ConcurrentLinkedQueue<>(Collections.singleton(startPage));
   Queue<Page> parsedPages = new ConcurrentLinkedQueue<>();
   Set<String> receivedLinks = new ConcurrentSkipListSet<>();

   var executor = Executors.newCachedThreadPool();

   do {
       Page currentPage = rawPages.poll();
       if (currentPage != null) {
           executor.execute(makeSearch(currentPage, rawPages, parsedPages, receivedLinks, endPageTitle));
       }
   } while (!receivedLinks.contains(endPageTitle));
   executor.shutdown();
   executor.close();

   parsedPages.add(
           new Page(endPageTitle,
                   rawPages.stream()
                           .filter(p -> p.getTitle().equals(endPageTitle))
                           .findAny()
                           .orElseThrow()
           )
   );

   return getResultPath(parsedPages, endPageTitle);
}
```

```java
@Override
public List<String> play(@NotNull String startPageTitle, @NotNull String endPageTitle, int maxDepth) {
    var startPage = new Page(startPageTitle, null);
    Queue<Page> rawPages = new ConcurrentLinkedQueue<>(Collections.singleton(startPage));
    Queue<Page> parsedPages = new ConcurrentLinkedQueue<>();
    Set<String> receivedLinks = new ConcurrentSkipListSet<>();

    var executor = Executors.newCachedThreadPool();

    do {
        Page currentPage = rawPages.poll();
        if (currentPage != null) {
            executor.execute(makeSearch(currentPage, rawPages, parsedPages, receivedLinks, endPageTitle));
        }
    } while (!receivedLinks.contains(endPageTitle));
    executor.shutdown();
    executor.close();

    parsedPages.add(
            new Page(endPageTitle,
                    rawPages.stream()
                            .filter(p -> p.getTitle().equals(endPageTitle))
                            .findAny()
                            .orElseThrow()
            )
    );

    return getResultPath(parsedPages, endPageTitle);
}
```

```java
@Override
public List<String> play(@NotNull String startPageTitle, @NotNull String endPageTitle, int maxDepth) {
    var startPage = new Page(startPageTitle, null);
    Queue<Page> rawPages = new ConcurrentLinkedQueue<>(Collections.singleton(startPage));
    Queue<Page> parsedPages = new ConcurrentLinkedQueue<>();
    Set<String> receivedLinks = new ConcurrentSkipListSet<>();

    var executor = Executors.newCachedThreadPool();

    do {
        Page currentPage = rawPages.poll();
        if (currentPage != null) {
            executor.execute(makeSearch(currentPage, rawPages, parsedPages, receivedLinks, endPageTitle));
        }
    } while (!receivedLinks.contains(endPageTitle));
    executor.shutdown();
    executor.close();

    parsedPages.add(
            new Page(endPageTitle,
                    rawPages.stream()
                            .filter(p -> p.getTitle().equals(endPageTitle))
                            .findAny()
                            .orElseThrow()
            )
    );

    return getResultPath(parsedPages, endPageTitle);
}
```

```java
@Override
public List<String> play(@NotNull String startPageTitle, @NotNull String endPageTitle, int maxDepth) {
    var startPage = new Page(startPageTitle, null);
    Queue<Page> rawPages = new ConcurrentLinkedQueue<>(Collections.singleton(startPage));
    Queue<Page> parsedPages = new ConcurrentLinkedQueue<>();
    Set<String> receivedLinks = new ConcurrentSkipListSet<>();

    var executor = Executors.newCachedThreadPool();

    do {
        Page currentPage = rawPages.poll();
        if (currentPage != null) {
            executor.execute(makeSearch(currentPage, rawPages, parsedPages, receivedLinks, endPageTitle));
        }
    } while (!receivedLinks.contains(endPageTitle));
    executor.shutdown();
    executor.close();

    parsedPages.add(
            new Page(endPageTitle,
                    rawPages.stream()
                            .filter(p -> p.getTitle().equals(endPageTitle))
                            .findAny()
                            .orElseThrow()
            )
    );

    return getResultPath(parsedPages, endPageTitle);
}
```

```java
@Override
public List<String> play(@NotNull String startPageTitle, @NotNull String endPageTitle, int maxDepth) {
    var startPage = new Page(startPageTitle, null);
    Queue<Page> rawPages = new ConcurrentLinkedQueue<>(Collections.singleton(startPage));
    Queue<Page> parsedPages = new ConcurrentLinkedQueue<>();
    Set<String> receivedLinks = new ConcurrentSkipListSet<>();

    var executor = Executors.newCachedThreadPool();

    do {
        Page currentPage = rawPages.poll();
        if (currentPage != null) {
            executor.execute(makeSearch(currentPage, rawPages, parsedPages, receivedLinks, endPageTitle));
        }
    } while (!receivedLinks.contains(endPageTitle));
    executor.shutdown();
    executor.close();

    parsedPages.add(
            new Page(endPageTitle,
                    rawPages.stream()
                            .filter(p -> p.getTitle().equals(endPageTitle))
                            .findAny()
                            .orElseThrow()
            )
    );

    return getResultPath(parsedPages, endPageTitle);
}
```

```
public Runnable makeSearch(Page currentPage, Queue<Page> rawPages, Queue<Page> parsedPages, Set<String>
receivedLinks, String endPageTitle) {
    return () -> {
        if (currentPage != null) {
            List<String> newLinks;
            try {
                newLinks = wikiDataSource.getLinksByTitle(currentPage.getTitle());
            } catch (Throwable e) {
                newLinks = null;
            }

            if (newLinks != null) {
                parsedPages.add(currentPage);
                for (String link : newLinks) {
                    if (receivedLinks.add(link)) {
                        rawPages.add(new Page(link, currentPage));
                    }
                }
            } else {
                rawPages.add(currentPage);
            }
        }
    };
}
```

```
3,014 ms    do {
                Page curPage = rawPages.poll();
                if (curPage != null) {
9,191 ms            exec.execute(makeSearch(curPage, rawPages, parsedPages, receivedLinks));
                }

6,251 ms    } while (!receivedLinks.contains(endPageTitle));
```

```
public Runnable makeSearch(Queue<Page> rawPages, Queue<Page> parsedPages, Set<String> receivedLinks,
String endPageTitle) {
    return () -> {
        Page currentPage = rawPages.poll();
        if (currentPage != null) {
            List<String> newLinks;
            try {
                newLinks = wikiDataSource.getLinksByTitle(currentPage.getTitle());
            } catch (Throwable e) {
                newLinks = null;
            }

            if (newLinks != null) {
                parsedPages.add(currentPage);
                for (String link : newLinks) {
                    if (receivedLinks.add(link)) {
                        rawPages.add(new Page(link, currentPage));
                    }
                    if (endPageTitle.equals(link)) {
                        isFinished.set(true);
                        break;
                    }
                }
            } else {
                rawPages.add(currentPage);
            }
        }
    };
}
```

```java
@Override
public List<String> play(@NotNull String startPageTitle, @NotNull String endPageTitle, int maxDepth) {
    var startPage = new Page(startPageTitle, null);
    Queue<Page> rawPages = new ConcurrentLinkedQueue<>(Collections.singleton(startPage));
    Queue<Page> parsedPages = new ConcurrentLinkedQueue<>();
    Set<String> receivedLinks = new ConcurrentSkipListSet<>();

    var executor = Executors.newCachedThreadPool();

    do {
        executor.execute(makeSearch(rawPages, parsedPages, receivedLinks, endPageTitle));
    } while (!isFinished.get());
    executor.shutdown();
    executor.close();

    parsedPages.add(
            new Page(endPageTitle,
                    rawPages.stream()
                            .filter(p -> p.getTitle().equals(endPageTitle))
                            .findAny()
                            .orElseThrow()
            )
    );

    return getResultPath(parsedPages, endPageTitle);
}
```

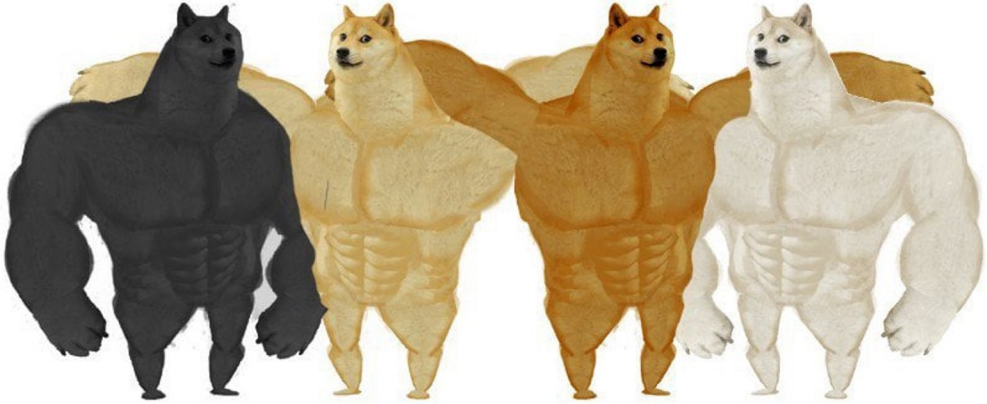
Есть нюанс

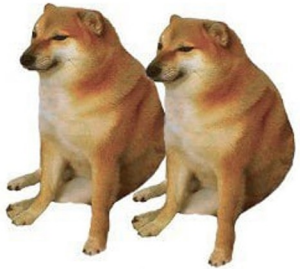
- Ограничение API Википедии – 200 RPS
- Воспользуемся Rate Limiter (resilience4j)

Замеры: Executors

Начальная страница	Конечная страница	Школьники	Executors
Бакуган	Библия	0,73 с.	5,22 сек.
!!!	Теория гомологий	0,87 с.	314,29 сек. // 5,2 мин.
А (кириллица)	Йод	0,84 с.	34,1 сек.
!!!!!!!	Физика	0,76 с.	20,05 сек.
Бутерброд	Бродский	0,53 с.	20,77 сек.
Охотники за привидениями	Пуджа	0,75 с.	46,19 сек.

Школьники





Мы

Executors – лучше, но не сильно

- Код не очень
- Сильно быстрее, чем синхронное решение, но медленнее школьников
- Надо лучше!

Completable Future

Kotlin Coroutines

Kotlin Coroutines

- Ключевое слово `suspend`
- Пишем последовательный код
- Асинхронные функции преобразуются компилятором в CPS

Kotlin Coroutines

```
interface WikiRepository {
    suspend fun getLinksByTitle(title: String): List<String>
}

interface WikiDataSource {
    suspend fun getLinksByTitle(title: String): List<String>
}
```

Реализации

```
class WikiRepositoryImpl(private val wikiDataSource: WikiDataSource) : WikiRepository {
    override suspend fun getLinksByTitle(title: String): List<String> =
        wikiDataSource.getLinksByTitle(title)
}
```

```kotlin
class WikiDataSourceImpl : WikDataSource {

    /* URL, Rate Limiter, HTTP Client */

    override suspend fun getLinksByTitle(title: String): List<String> {
        val response = rateLimiter.executeSuspendFunction {
            client.get(URL) { /* Parameters */ }
        }

        val wikiLinkResponse: WikiLinksResponse = response.body()

        val links = /* Get links from wikiLinksResponse */

        return links
    }
}
```

```
override fun play(
    startPageTitle: String,
    endPageTitle: String,
    maxDepth: Int
): List<String> = runBlocking {
    val visitedPages: MutableMap<String, Boolean> = ConcurrentHashMap()

    val startPage = Page(startPageTitle, null)

    val resultPage = processPage(startPage, endPageTitle, 0, maxDepth, visitedPages)

    val path = /* Get path from resultPage */

    return@runBlocking path
}
```

```
override fun play(
    startPageTitle: String,
    endPageTitle: String,
    maxDepth: Int
): List<String> = runBlocking {
    val visitedPages: MutableMap<String, Boolean> = ConcurrentHashMap()

    val startPage = Page(startPageTitle, null)

    val resultPage = processPage(startPage, endPageTitle, 0, maxDepth, visitedPages)

    val path = /* Get path from resultPage */

    return@runBlocking path
}
```

```
override fun play(
    startPageTitle: String,
    endPageTitle: String,
    maxDepth: Int
): List<String> = runBlocking {
    val visitedPages: MutableMap<String, Boolean> = ConcurrentHashMap()

    val startPage = Page(startPageTitle, null)

    val resultPage = processPage(startPage, endPageTitle, 0, maxDepth, visitedPages)

    val path = /* Get path from resultPage */

    return@runBlocking path
}
```

```
override fun play(
    startPageTitle: String,
    endPageTitle: String,
    maxDepth: Int
): List<String> = runBlocking {
    val visitedPages: MutableMap<String, Boolean> = ConcurrentHashMap()

    val startPage = Page(startPageTitle, null)

    val resultPage = processPage(startPage, endPageTitle, 0, maxDepth, visitedPages)

    val path = /* Get path from resultPage */

    return@runBlocking path
}
```

```
override fun play(
    startPageTitle: String,
    endPageTitle: String,
    maxDepth: Int
): List<String> = runBlocking {
    val visitedPages: MutableMap<String, Boolean> = ConcurrentHashMap()

    val startPage = Page(startPageTitle, null)

    val resultPage = processPage(startPage, endPageTitle, 0, maxDepth, visitedPages)

    val path = /* Get path from resultPage */

    return@runBlocking path
}
```

```
private suspend fun processPage(
    page: Page,
    endPageTitle: String,
    curDepth: Int,
    maxDepth: Int,
    visitedPages: MutableMap<String, Boolean>,
): Page
```

Проверяем граничные условия...

```
if (visitedPages.putIfAbsent(page.title, true) != null) {
    throw RuntimeException("Already visited")
}

if (page.title == endPageTitle) {
    return page
}

if (curDepth == maxDepth) {
    throw RuntimeException("Depth reached")
}
```

Проверяем граничные условия...

```
if (visitedPages.putIfAbsent(page.title, true) != null) {
    throw RuntimeException("Already visited")
}

if (page.title == endPageTitle) {
    return page
}

if (curDepth == maxDepth) {
    throw RuntimeException("Depth reached")
}
```

Проверяем граничные условия...

```
if (visitedPages.putIfAbsent(page.title, true) != null) {
    throw RuntimeException("Already visited")
}

if (page.title == endPageTitle) {
    return page
}

if (curDepth == maxDepth) {
    throw RuntimeException("Depth reached")
}
```

Проверяем граничные условия...

```
if (visitedPages.putIfAbsent(page.title, true) != null) {
    throw RuntimeException("Already visited")
}

if (page.title == endPageTitle) {
    return page
}

if (curDepth == maxDepth) {
    throw RuntimeException("Depth reached")
}
```

...получаем новые ссылки и продолжаем поиск

```
val links = wikiRepository.getLinksByTitle(page.title)

val pageChannel = Channel<Page>()

val scope = CoroutineScope(SupervisorJob() + CoroutineExceptionHandler { _, _ -> })
links.forEach { link ->
    scope.launch {
        val pageResult = processPage(
            Page(link, page),
            endPageTitle,
            curDepth + 1,
            maxDepth,
            visitedPages,
        )

        pageChannel.send(pageResult)
    }
}

val resultPage = pageChannel.receive()
scope.cancel()

return resultPage
```

...получаем новые ссылки и продолжаем поиск

```
val links = wikiRepository.getLinksByTitle(page.title)

val pageChannel = Channel<Page>()

val scope = CoroutineScope(SupervisorJob() + CoroutineExceptionHandler { _, _ -> })
links.forEach { link ->
    scope.launch {
        val pageResult = processPage(
            Page(link, page),
            endPageTitle,
            curDepth + 1,
            maxDepth,
            visitedPages,
        )

        pageChannel.send(pageResult)
    }
}

val resultPage = pageChannel.receive()
scope.cancel()

return resultPage
```

...получаем новые ссылки и продолжаем поиск

```
val links = wikiRepository.getLinksByTitle(page.title)

val pageChannel = Channel<Page>()

val scope = CoroutineScope(SupervisorJob() + CoroutineExceptionHandler { _, _ -> })
links.forEach { link ->
    scope.launch {
        val pageResult = processPage(
            Page(link, page),
            endPageTitle,
            curDepth + 1,
            maxDepth,
            visitedPages,
        )

        pageChannel.send(pageResult)
    }
}

val resultPage = pageChannel.receive()
scope.cancel()

return resultPage
```

...получаем новые ссылки и продолжаем поиск

```
val links = wikiRepository.getLinksByTitle(page.title)

val pageChannel = Channel<Page>()

val scope = CoroutineScope(SupervisorJob() + CoroutineExceptionHandler { _, _ -> })
links.forEach { link ->
    scope.launch {
        val pageResult = processPage(
            Page(link, page),
            endPageTitle,
            curDepth + 1,
            maxDepth,
            visitedPages,
        )

        pageChannel.send(pageResult)
    }
}

val resultPage = pageChannel.receive()
scope.cancel()

return resultPage
```

...получаем новые ссылки и продолжаем поиск

```
val links = wikiRepository.getLinksByTitle(page.title)

val pageChannel = Channel<Page>()

val scope = CoroutineScope(SupervisorJob() + CoroutineExceptionHandler { _, _ -> })
links.forEach { link ->
    scope.launch {
        val pageResult = processPage(
            Page(link, page),
            endPageTitle,
            curDepth + 1,
            maxDepth,
            visitedPages,
        )

        pageChannel.send(pageResult)
    }
}

val resultPage = pageChannel.receive()
scope.cancel()

return resultPage
```

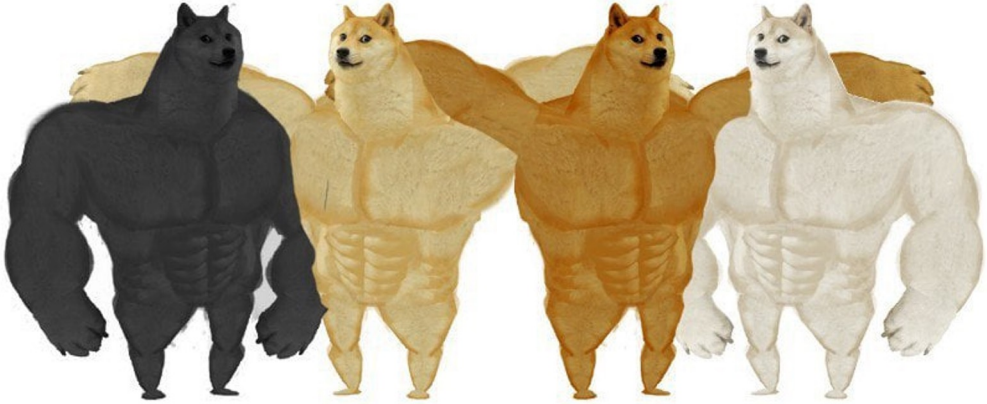
...получаем новые ссылки и продолжаем поиск

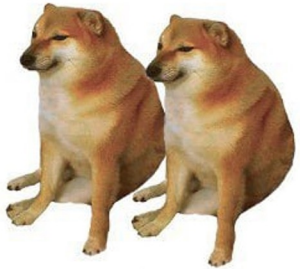
```
val links = wikiRepository.getLinksByTitle(page.title)

val pageChannel = Channel<Page>()

val scope = CoroutineScope(SupervisorJob() + CoroutineExceptionHandler { _, _ -> })
links.forEach { link ->
    scope.launch {
        val pageResult = processPage(
            Page(link, page),
            endPageTitle,
            curDepth + 1,
            maxDepth,
            visitedPages,
        )

        pageChannel.send(pageResult)
    }
}

val resultPage = pageChannel.receive()
scope.cancel()

return resultPage
```

Замеры: Kotlin Coroutines

Начальная страница	Конечная страница	Школьники	Coroutines
Бакуган	Библия	0,73 с.	7,53 с.
!!!	Теория гомологий	0,87 с.	466,65 с. *// 7,7 м.*
А (кириллица)	Йод	0,84 с.	35,38 с.
!!!!!!!	Физика	0,76 с.	27,45 с.
Бутерброд	Бродский	0,53 с.	111,6 с. *// 1,8 м.*
Охотники за привидениями	Пуджа	0,75 с.	947,58 с. *// 15,8 м.*

Школьники





Мы

Что не так?

- Код стал лучше
- Писать и читать приятнее, менять проще

Что не так?

- Код стал лучше
- Писать и читать приятнее, менять проще
- Производительность перебора страниц стала на порядок лучше

Что не так?

- Код стал лучше
- Писать и читать приятнее, менять проще
- Производительность перебора страниц стала на порядок лучше
- Но скорость все еще не очень…

Что не так?

- Код стал лучше
- Писать и читать приятнее, менять проще
- Производительность перебора страниц стала на порядок лучше
- Но скорость все еще не очень…
- Может, взять что-нибудь круче и новее?

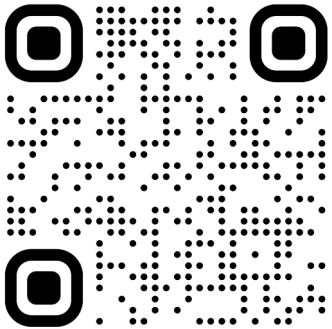
Project Loom

Loom

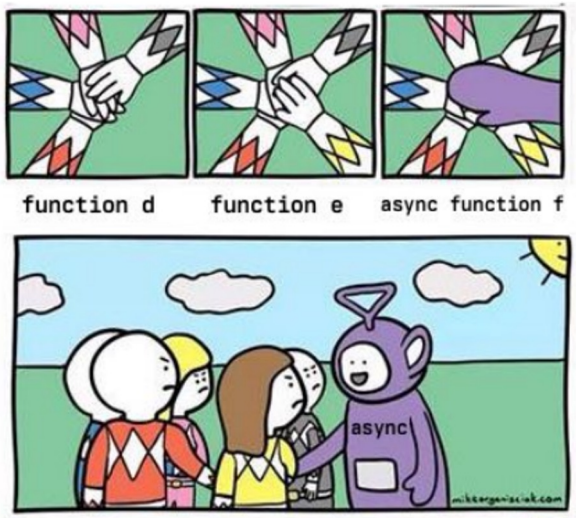
- Вошел в состав 21 JDK – релиз 21 день назад
- Виртуальные потоки
- Structured Concurrency

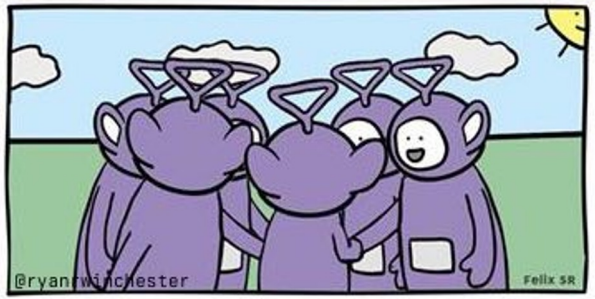
Loom

- Вошел в состав 21 JDK – релиз 21 день назад
- Виртуальные потоки
- Structured Concurrency
- Не делим мир на “цвета” (Bob Nystrom, “What Color is Your Function?”)




function a
function b
function c


function d
function e
async function f
async


@ryanrwinchester
Felix SR

Loom

- Вошел в состав 21 JDK – релиз 21 день назад
- Виртуальные потоки
- Structured Concurrency
- Не делим мир на “цвета” (Bob Nystrom, “What Color is Your Function?”)
- Возвращаемся в мир Java

```java
@Override
public List<String> play(String startPageTitle, String endPageTitle, int maxDepth) {
    var visitedPages = new ConcurrentHashMap<String, Boolean>();

    var startPage = new Page(startPageTitle, null);

    Page resultPage = processPage(startPage, endPageTitle, 0, maxDepth, visitedPages);

    var path = /* Get path from resultPage */;
    return path;
}
```

```java
@Override
public List<String> play(String startPageTitle, String endPageTitle, int maxDepth) {
    var visitedPages = new ConcurrentHashMap<String, Boolean>();

    var startPage = new Page(startPageTitle, null);

    Page resultPage = processPage(startPage, endPageTitle, 0, maxDepth, visitedPages);

    var path = /* Get path from resultPage */;
    return path;
}
```

```java
@Override
public List<String> play(String startPageTitle, String endPageTitle, int maxDepth) {
    var visitedPages = new ConcurrentHashMap<String, Boolean>();

    var startPage = new Page(startPageTitle, null);

    Page resultPage = processPage(startPage, endPageTitle, 0, maxDepth, visitedPages);

    var path = /* Get path from resultPage */;
    return path;
}
```

```java
@Override
public List<String> play(String startPageTitle, String endPageTitle, int maxDepth) {
    var visitedPages = new ConcurrentHashMap<String, Boolean>();

    var startPage = new Page(startPageTitle, null);

    Page resultPage = processPage(startPage, endPageTitle, 0, maxDepth, visitedPages);

    var path = /* Get path from resultPage */;
    return path;
}
```

```java
@Override
public List<String> play(String startPageTitle, String endPageTitle, int maxDepth) {
    var visitedPages = new ConcurrentHashMap<String, Boolean>();

    var startPage = new Page(startPageTitle, null);

    Page resultPage = processPage(startPage, endPageTitle, 0, maxDepth, visitedPages);

    var path = /* Get path from resultPage */;
    return path;
}
```

```
private Page processPage(
    Page page,
    String endPageTitle,
    int curDepth,
    int maxDepth,
    Map<String, Boolean> visitedPages
)
```

Проверяем граничные условия...

```
if (visitedPages.putIfAbsent(page.getTitle(), true) != null) {
    throw new RuntimeException("Already visited");
}

if (page.getTitle().equals(endPageTitle)) {
    return page;
}

if (curDepth == maxDepth) {
    throw new RuntimeException("Depth reached");
}
```

...получаем новые ссылки и продолжаем поиск

```
var links = wikiRepository.getLinksByTitle(page.getTitle());
try (var scope = new StructuredTaskScope.ShutdownOnSuccess<Page>()) {
    links.forEach((link) -> { scope.fork(() -> processPage(
                new Page(link, page),
                endPageTitle,
                curDepth + 1,
                maxDepth,
                visitedPages
            )
        );
    });
    scope.join();

    return scope.result();
} catch (Throwable e) {
    throw new RuntimeException(e);
}
```

...получаем новые ссылки и продолжаем поиск

```
var links = wikiRepository.getLinksByTitle(page.getTitle());
try (var scope = new StructuredTaskScope.ShutdownOnSuccess<Page>()) {
    links.forEach((link) -> { scope.fork(() -> processPage(
                new Page(link, page),
                endPageTitle,
                curDepth + 1,
                maxDepth,
                visitedPages
            )
        );
    });
    scope.join();

    return scope.result();
} catch (Throwable e) {
    throw new RuntimeException(e);
}
```

...получаем новые ссылки и продолжаем поиск

```
var links = wikiRepository.getLinksByTitle(page.getTitle());
try (var scope = new StructuredTaskScope.ShutdownOnSuccess<Page>()) {
    links.forEach((link) -> { scope.fork(() -> processPage(
                new Page(link, page),
                endPageTitle,
                curDepth + 1,
                maxDepth,
                visitedPages
            )
        );
    });
    scope.join();

    return scope.result();
} catch (Throwable e) {
    throw new RuntimeException(e);
}
```

...получаем новые ссылки и продолжаем поиск

```
var links = wikiRepository.getLinksByTitle(page.getTitle());
try (var scope = new StructuredTaskScope.ShutdownOnSuccess<Page>()) {
    links.forEach((link) -> { scope.fork(() -> processPage(
                new Page(link, page),
                endPageTitle,
                curDepth + 1,
                maxDepth,
                visitedPages
            )
        );
    });
    scope.join();

    return scope.result();
} catch (Throwable e) {
    throw new RuntimeException(e);
}
```

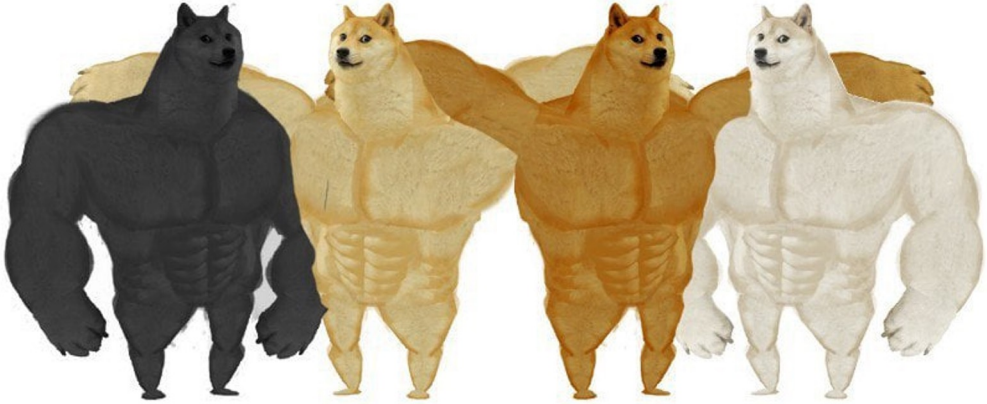
...получаем новые ссылки и продолжаем поиск

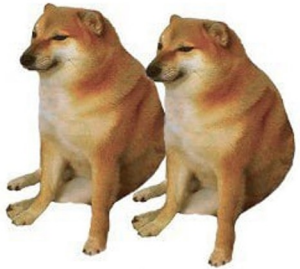
```
var links = wikiRepository.getLinksByTitle(page.getTitle());
try (var scope = new StructuredTaskScope.ShutdownOnSuccess<Page>()) {
    links.forEach((link) -> { scope.fork(() -> processPage(
                new Page(link, page),
                endPageTitle,
                curDepth + 1,
                maxDepth,
                visitedPages
            )
        );
    });
    scope.join();

    return scope.result();
} catch (Throwable e) {
    throw new RuntimeException(e);
}
```

Замеры: Loom

Начальная страница	Конечная страница	Школьники	Loom
Бакуган	Библия	0,73 с.	5,8 с.
!!!	Теория гомологий	0,87 с.	508,02 с. *// 8,5 м.*
А (кириллица)	Йод	0,84 с.	35,76 с.
!!!!!!!	Физика	0,76 с.	37,76 с.
Бутерброд	Бродский	0,53 с.	104,6 с. *// 1,7 м.*
Охотники за привидениями	Пуджа	0,75 с.	977,92 с. *// 16,3 м.*

Школьники





Мы


1:1
0 ms
15 sec
30 sec
45 sec
1 min
HttpClient-1-Worker-0
HttpClient-1-Worker-1
HttpClient-1-SelectorManager
RMI TCP Connection(idle)
ForkJoinPool-1-worker-8
ForkJoinPool-1-worker-9
ForkJoinPool-1-worker-10
ForkJoinPool-1-worker-11
ForkJoinPool-1-worker-12
ForkJoinPool-1-worker-1
ForkJoinPool-1-worker-2
ForkJoinPool-1-worker-3
ForkJoinPool-1-worker-4
ForkJoinPool-1-worker-5
ForkJoinPool-1-worker-6
ForkJoinPool-1-worker-7

Что не так?

- Взяли самую современную технологию

Что не так?

- Взяли самую современную технологию
- Ожидали, что она сделает нашу жизнь лучше и ускорит решение
- Но не помогло

Что не так?

- Взяли самую современную технологию
- Ожидали, что она сделает нашу жизнь лучше и ускорит решение
- Но не помогло
- Грусть, печаль. Чем это лучше Executors?

Пытаемся разобраться

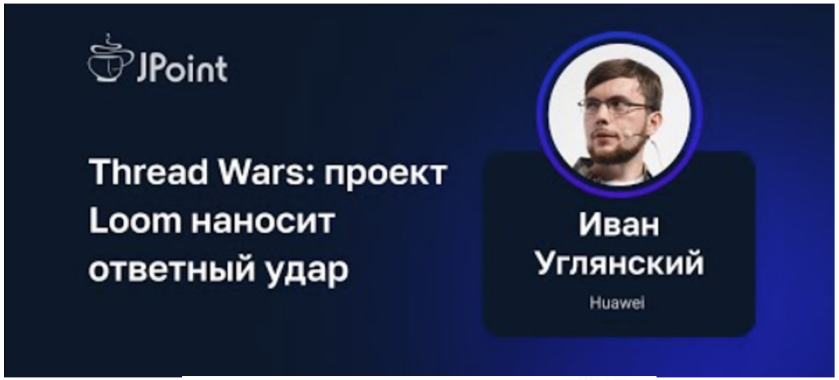
- Гуглим, заходим в ютуб

Пытаемся разобраться

- Гуглим, заходим в ютуб
- Находим доклад Вани Углянского

Пытаемся разобраться

- Гуглим, заходим в ютуб
- Находим доклад Вани Углянского



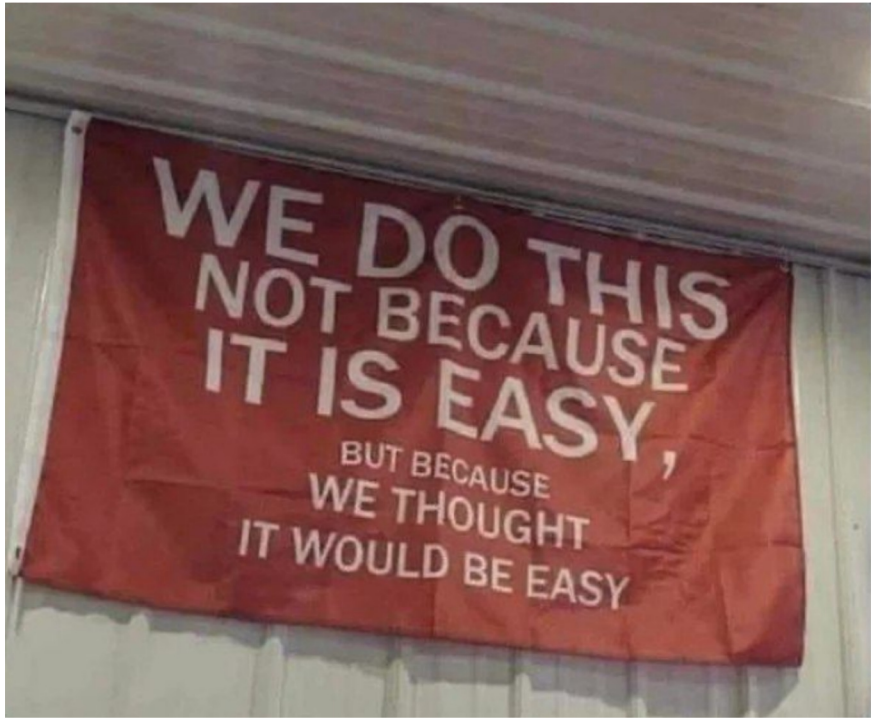
Пытаемся разобраться

- Гуглим, заходим в ютуб
- Находим доклад Вани Углянского
- Понимаем, что все подходы к асинхронности фундаментально основаны на одних и тех же вещах

Пытаемся разобраться

- Гуглим, заходим в ютуб
- Находим доклад Вани Углянского
- Понимаем, что все подходы к асинхронности фундаментально основаны на одних и тех же вещах
- Но они позволяют нам писать более простой, лаконичный и понятный код, который меньше подвержен ошибкам

А что делать то?

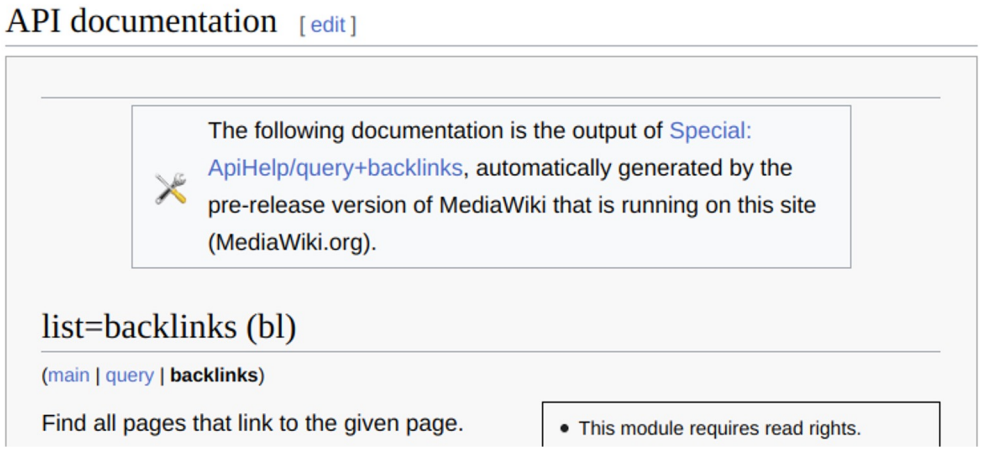

WE DO THIS
NOT BECAUSE
IT IS EASY,
BUT BECAUSE
WE THOUGHT
IT WOULD BE EASY

Решаем задачу

- Наша проблема не в ожидании запросов, а в количестве страниц

Решаем задачу

- Наша проблема не в ожидании запросов, а в количестве страниц
- Идем в API Википедии
- Находим backlinks



Решаем задачу

- Наша проблема не в ожидании запросов, а в количестве страниц
- Идем в API Википедии
- Находим backlinks
- Возникает ~~очень сложная~~ идея

Решаем задачу

- Наша проблема не в ожидании запросов, а в количестве страниц
- Идем в API Википедии
- Находим backlinks
- Возникает ~~очень сложная~~ идея
- BFS с двух сторон!

Two-Way BFS

Расширим интерфейсы

```
public interface WikiRepository {
    List<String> getLinksByTitle(String title);
    List<String> getBacklinksByTitle(String title);
}

public interface WikiDataSource {
    List<String> getLinksByTitle(String title);
    List<String> getBacklinksByTitle(String title);
}
```

```java
@Override
public List<String> play(String startPageTitle, String endPageTitle, int maxDepth) {
    var visitedForwardPages = new ConcurrentHashMap<String, ForwardPage>();
    var visitedBackwardPages = new ConcurrentHashMap<String, BackwardPage>();

    var startForwardPage = new ForwardPage(startPageTitle, null);
    var endBackwardPage = new BackwardPage(endPageTitle, null);

    ...
```

```
...
try (var scope = new StructuredTaskScope.ShutdownOnSuccess<PairPages>()) {
    scope.fork(() -> processPageForward(
        startForwardPage,
        0,
        maxDepth,
        visitedForwardPages,
        visitedBackwardPages
    ));

    scope.fork(() -> processPageBackward(
        endBackwardPage,
        0,
        maxDepth,
        visitedForwardPages,
        visitedBackwardPages
    ));

    scope.join();

    var pairPagesResult = scope.result();

    return getFinalPathFromForwardAndBackward(pairPagesResult.forwardPage, pairPagesResult.backwardPage);
} catch (Throwable e) {
    throw new RuntimeException(e);
}
```

```
...
try (var scope = new StructuredTaskScope.ShutdownOnSuccess<PairPages>()) {
    scope.fork(() -> processPageForward(
        startForwardPage,
        0,
        maxDepth,
        visitedForwardPages,
        visitedBackwardPages
    ));

    scope.fork(() -> processPageBackward(
        endBackwardPage,
        0,
        maxDepth,
        visitedForwardPages,
        visitedBackwardPages
    ));

    scope.join();

    var pairPagesResult = scope.result();

    return getFinalPathFromForwardAndBackward(pairPagesResult.forwardPage, pairPagesResult.backwardPage);
} catch (Throwable e) {
    throw new RuntimeException(e);
}
```

```
...
try (var scope = new StructuredTaskScope.ShutdownOnSuccess<PairPages>()) {
    scope.fork(() -> processPageForward(
        startForwardPage,
        0,
        maxDepth,
        visitedForwardPages,
        visitedBackwardPages
    ));

    scope.fork(() -> processPageBackward(
        endBackwardPage,
        0,
        maxDepth,
        visitedForwardPages,
        visitedBackwardPages
    ));

    scope.join();

    var pairPagesResult = scope.result();

    return getFinalPathFromForwardAndBackward(pairPagesResult.forwardPage, pairPagesResult.backwardPage);
} catch (Throwable e) {
    throw new RuntimeException(e);
}
```

```
...
try (var scope = new StructuredTaskScope.ShutdownOnSuccess<PairPages>()) {
    scope.fork(() -> processPageForward(
        startForwardPage,
        0,
        maxDepth,
        visitedForwardPages,
        visitedBackwardPages
    ));

    scope.fork(() -> processPageBackward(
        endBackwardPage,
        0,
        maxDepth,
        visitedForwardPages,
        visitedBackwardPages
    ));

    scope.join();

    var pairPagesResult = scope.result();

    return getFinalPathFromForwardAndBackward(pairPagesResult.forwardPage, pairPagesResult.backwardPage);
} catch (Throwable e) {
    throw new RuntimeException(e);
}
```

```
private record PairPages(ForwardPage forwardPage, BackwardPage backwardPage) {}
```

```
private record PairPages(ForwardPage forwardPage, BackwardPage backwardPage) {}

private PairPages processPageForward(
    ForwardPage page,
    int curDepth,
    int maxDepth,
    ConcurrentMap<String, ForwardPage> visitedForwardPages,
    ConcurrentMap<String, BackwardPage> visitedBackwardPages
)
```

Проверяем граничные условия...

```
if (visitedForwardPages.putIfAbsent(page.getTitle(), page) != null) {
    throw new RuntimeException("Already visited");
}

var backwardPage = visitedBackwardPages.get(page.getTitle());
if (backwardPage != null) {
    return new PairPages(page, backwardPage);
}

if (curDepth == maxDepth) {
    throw new RuntimeException("Depth reached");
}
```

...получаем новые ссылки и продолжаем поиск

```
var links = wikiRepository.getLinksByTitle(page.getTitle());

try (var scope = new StructuredTaskScope.ShutdownOnSuccess<PairPages>()) {
   links.forEach((link) -> {
       scope.fork(() -> processPageForward(
               new ForwardPage(link, page),
               curDepth + 1,
               maxDepth,
               visitedForwardPages,
               visitedBackwardPages
       ));
   });
   scope.join();

   return scope.result();
} catch (Throwable e) {
   throw new RuntimeException(e);
}
```

```
private PairPages processPageBackward(
    BackwardPage page,
    int curDepth,
    int maxDepth,
    ConcurrentMap<String, ForwardPage> visitedForwardPages,
    ConcurrentMap<String, BackwardPage> visitedBackwardPages
)
```

Замеры: Loom + Two-Way BFS

Начальная страница	Конечная страница	Школьники	Loom + Two-Way BFS
Бакуган	Библия	0,73 с.	0,78 с.
!!!	Теория гомологий	0,87 с.	1,69 с.
A (кириллица)	Йод	0,84 с.	1,31 с.
!!!!!!!	Физика	0,76 с.	1 с.
Бутерброд	Бродский	0,53 с.	1,14 с.
Охотники за привидениями	Пуджа	0,75 с.	1,49 с.

Но есть нюанс

- При решении задачи было два зачета: честный и нечестный

Но есть нюанс

- При решении задачи было два зачета: честный и нечестный
- В честном надо было ходить только по сети, но они сразу саботировали его тем, что забивали сеть, мешая друг другу

Но есть нюанс

- При решении задачи было два зачета: честный и нечестный
- В честном надо было ходить только по сети, но они сразу саботировали его тем, что забивали сеть, мешая друг другу
- В нечестном зачете юные умы хакнули наши правила и выкачивали себе Википедию, выгружали ее в графовую БД, предрасчитывали пути

Но есть нюанс

- При решении задачи было два зачета: честный и нечестный
- В честном надо было ходить только по сети, но они сразу саботировали его тем, что забивали сеть, мешая друг другу
- В нечестном зачете юные умы хакнули наши правила и выкачивали себе Википедию, выгружали ее в графовую БД, предрасчитывали пути
- А это на минуточку 222kk связей, 7kk страниц и 28гб данных MySQL

Хотим еще быстрее!

- Выгрузим все в локальную базу данных
- Избавимся от Rate Limiter

```
public class WikiMySqlDataSourceImpl implements WikiDataSource {

    // Connection to MySQL DB
    Connection connection = MySQLConnectionBuilder.createConnectionPool(
            "jdbc:mysql://localhost:3306/wiki?user=root&password=123456789");

    /* Methods */
}
```

```java
@Override
public List<String> getLinksByTitle(String title) {
    try {
        var queryResult = connection.sendPreparedStatement(
                        "SELECT pl_title
                                FROM pagelinks
                                JOIN page ON page_id = pl_from
                                WHERE page_title = ? AND pl_namespace = 0;",
                        Collections.singletonList(title)
                ).join();

        return queryResult.getRows().stream()
                .map(row -> row.get(0))
                .map(bytes -> new String((byte[]) bytes, StandardCharsets.UTF_8))
                .toList();
    } catch (Throwable e) {
        throw new RuntimeException(e);
    }
}
```

```java
@Override
public List<String> getBacklinksByTitle(String title) {
    try {
        var queryResult = connection.sendPreparedStatement(
                        "SELECT page_title
                                FROM pagelinks
                                JOIN page ON page_id = pl_from
                                WHERE pl_title = ? AND pl_namespace = 0;",
                        Collections.singletonList(title)
                ).join();

        return queryResult.getRows().stream()
                .map(row -> row.get(0))
                .map(bytes -> new String((byte[]) bytes, StandardCharsets.UTF_8))
                .toList();
    } catch (Throwable e) {
        throw new RuntimeException(e);
    }
}
```

Замеры: Loom + Two-Way BFS + DB

Начальная страница	Конечная страница	Школьники	Loom + Two-Way BFS + DB
Бакуган	Библия	0,73 с.	> 30 мин.
!!!	Теория гомологий	0,87 с.	…
А (кириллица)	Йод	0,84 с.	…
!!!!!!!	Физика	0,76 с.	…
Бутерброд	Бродский	0,53 с.	…
Охотники за привидениями	Пуджа	0,75 с.	…

С чем боролись…

- База данных нас ни в чем не ограничивает

С чем боролись…

- База данных нас ни в чем не ограничивает
- Выгружаем **очень много** страниц
- Планировщик **кооперативный**, распределение времени нас не устраивает


15:48:54
23%
CPU
4068 MB
Heap Memory


0 ms
5 sec
10 sec
15 sec
20 sec
25 sec
30 sec
35 sec
ForkJoinPool-1-worker-7
ForkJoinPool-1-worker-8
ForkJoinPool-1-worker-9
ForkJoinPool-1-worker-10
ForkJoinPool-1-worker-11
ForkJoinPool-1-worker-12
ForkJoinPool-1-worker-1
ForkJoinPool-1-worker-2
ForkJoinPool-1-worker-3
ForkJoinPool-1-worker-4
ForkJoinPool-1-worker-5
ForkJoinPool-1-worker-6
db-sql-netty-thread-1

С чем боролись…

- База данных нас ни в чем не ограничивает
- Выгружаем **очень много** страниц
- Планировщик **кооперативный**, распределение времени нас не устраивает
- Rate Limiter, внезапно, нам наоборот помогает!

Замеры: Loom + Two-Way BFS + DB + Rate Limiter

Начальная страница	Конечная страница	Школьники	Loom + Two-Way BFS + DB + RL
Бакуган	Библия	0,73 с.	0,62 с.
!!!	Теория гомологий	0,87 с.	0,83 с.
А (кириллица)	Йод	0,84 с.	0,68 с.
!!!!!!!	Физика	0,76 с.	0,61 с.
Бутерброд	Бродский	0,53 с.	0,57 с.
Охотники за привидениями	Пуджа	0,75 с.	0,66 с.

Вывод

- Развитие технологий это не про скорость (упарываться в оптимизации мы умели всегда), а про качество кода

Вывод

- Развитие технологий это не про скорость (упарываться в оптимизации мы умели всегда), а про качество кода
- При этом ускоряемся мы сами, становимся производительнее

Вывод

- Развитие технологий это не про скорость (упарываться в оптимизации мы умели всегда), а про качество кода
- При этом ускоряемся мы сами, становимся производительнее
- Появляется больше времени на продумывание решений

ТИНЬКОФФ

Конкурс для WikiGame

ТИНЬКОФФ