

Инструменты микросервисной архитектуры

С. Ануфриев, Е. Штыков

Микро-что?

Веб-сервисы - это сервисы (программы), которые общаются между собой и окружающим миром по http.

Архитектура - способ организации этого общения в конкретном проекте.

Микро - когда набор методов у каждого сервиса невелик.

Инструменты

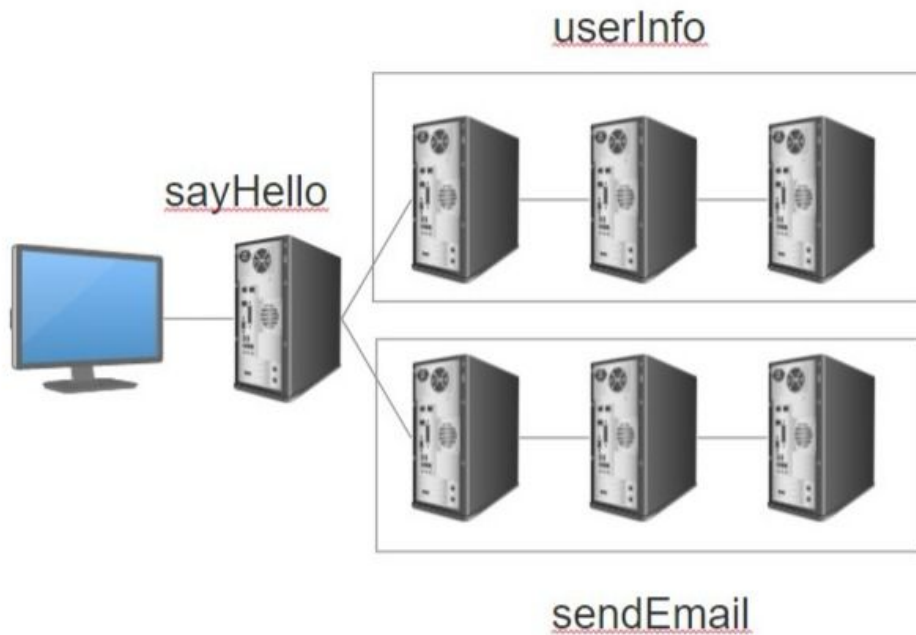
Кластер-клиент

- Направляет запрос на нужную реплику сервиса
- Настройки топологии меняются на лету
- Обновление сервиса без даунтайма

Распределенная трассировка

- Показывает дочерние запросы
- Собирает статистику по всем репликам
- Прямой доступ к логам

Архитектура



План разработки

Шаг 0

Пишем простейший эхо-сервис

sayHello

Шаг 2

Добавляем второй дочерний запрос

userInfo

sendEmail

Шаг 1

Вызываем другой сервис через кластер-клиент

Шаг 3

Смотрим трассировку запросов

Vert.x

Eclipse Vert.x is a toolkit for building reactive applications on the JVM.

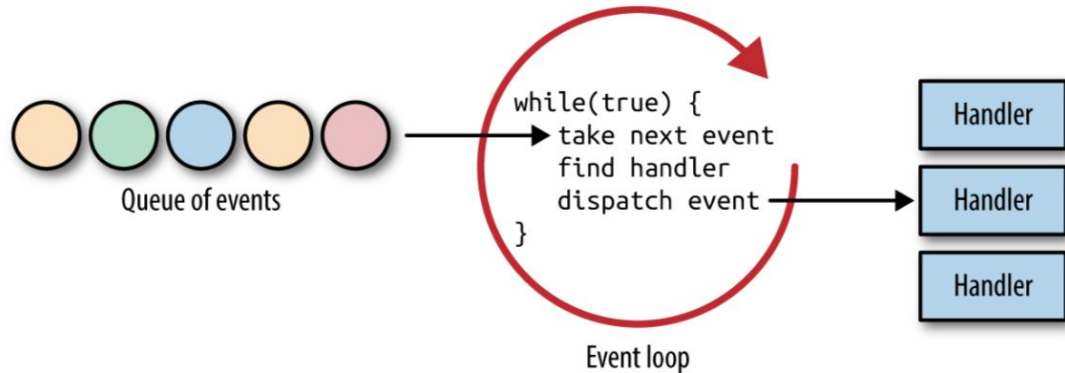
— Vert.x website

- **Легковесность**
- **Мультиязычность: Java, Scala, Kotlin, Groovy, Ceylon**
- **Асинхронность**
- **Java 8**

Модули Vert.x

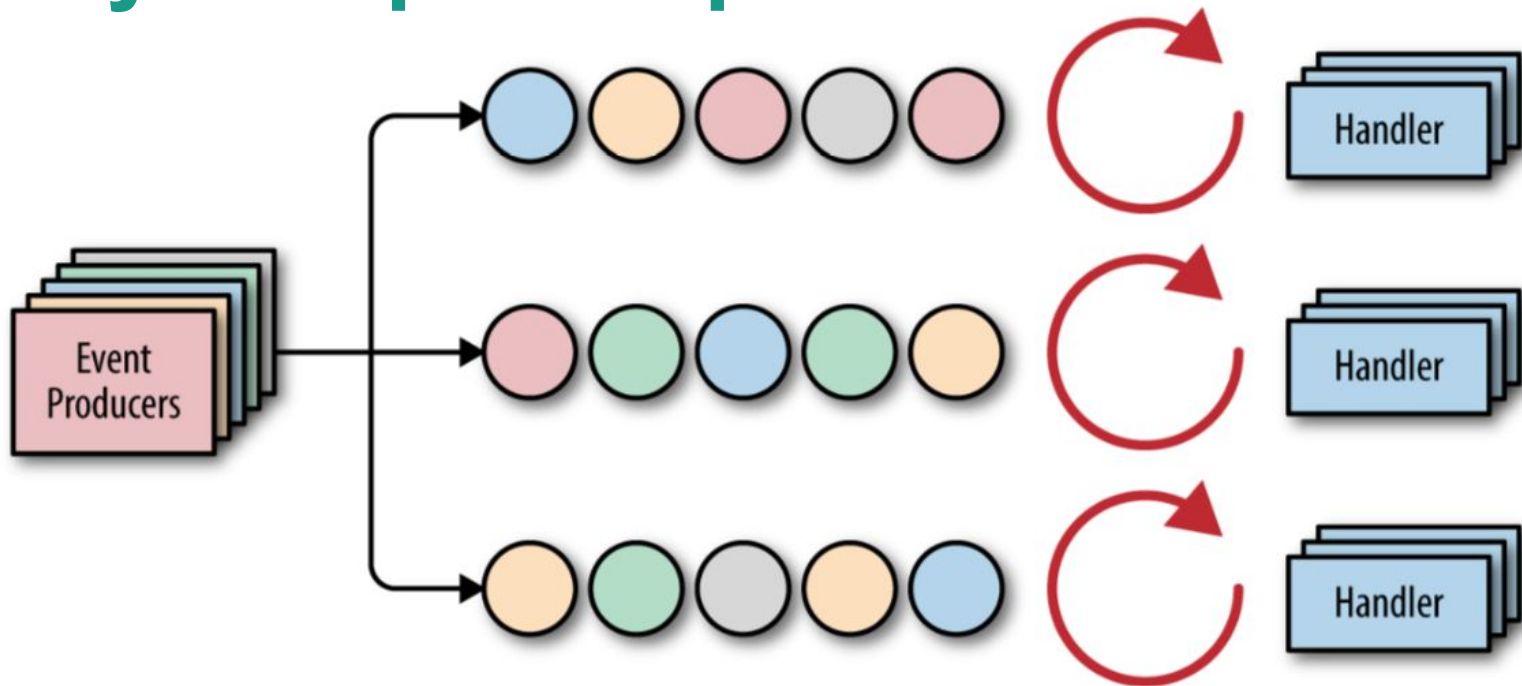
- Core: HTTP, TCP, file system access
- Web: построение web приложений
- Data access: MongoDB клиент, JDBC клиент.
- Devops: Metrics

Событийно-ориентированное построение приложения



```
vertx.createHttpServer()  
  .requestHandler(request -> {  
    // This handler will be called every time an HTTP  
    // request is received at the server  
    request.response().end("hello Vert.x");  
  })  
  .listen(8080);
```


Мультиреактор



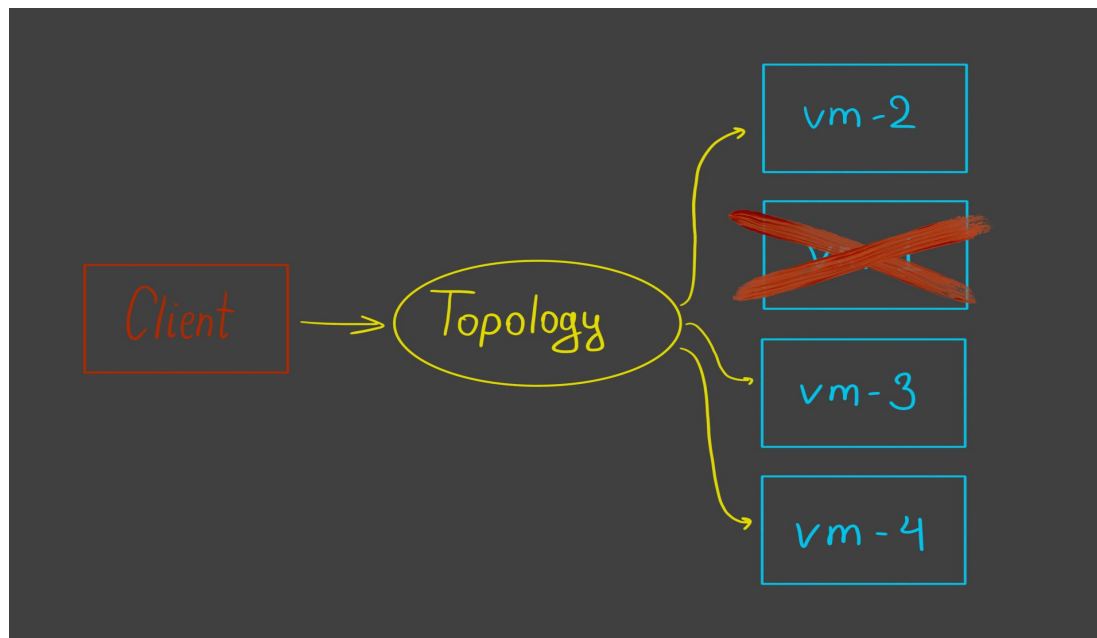


Шаг 0.

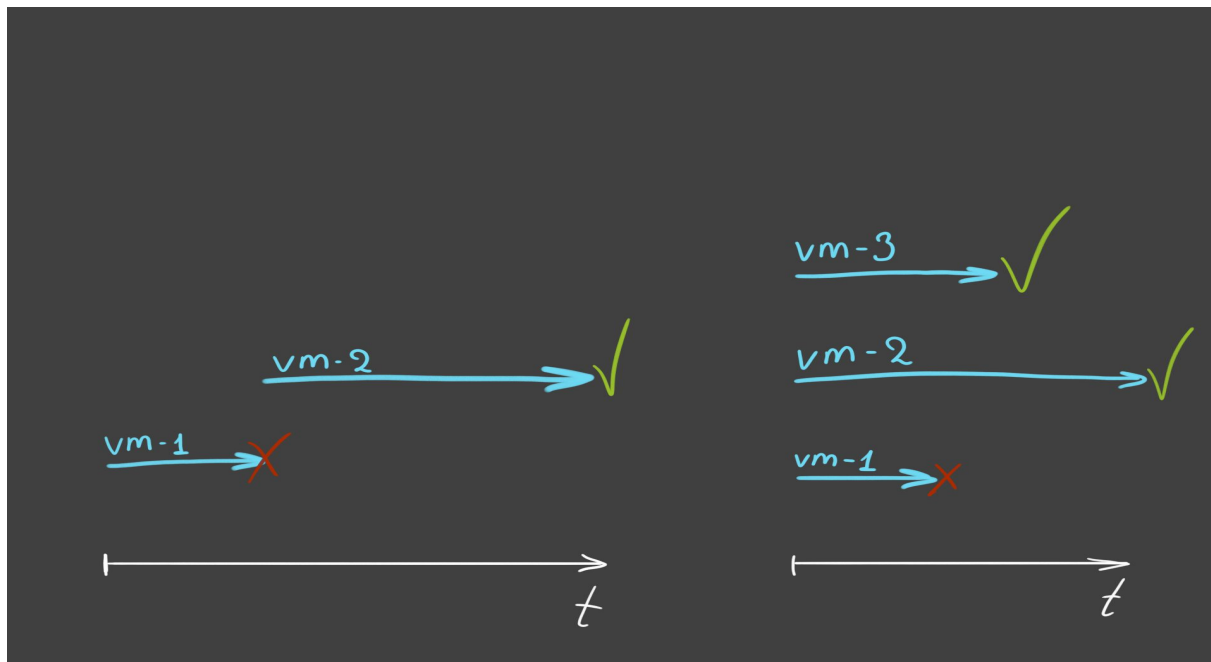
Сервис sayHello, который возвращает то, что прислали.

```
public void start() {  
    Router router = Router.router(vertex);  
  
    router.get("/say/hello/:id").handler(ctx -> {  
        String userId = ctx.request().getParam("id");  
  
        JsonObject response = new JsonObject().put(  
            "name", userName);  
        ctx.response().end(response.toBuffer());  
    });  
  
    vertex  
        .createHttpServer()  
        .requestHandler(router::accept)  
        .listen(8080);  
}
```

Кластер-клиент



Кластер-клиент





Шаг 1.

Вызов сервиса userInfo с помощью кластер-клиента и получение списка друзей.

```
IClusterClient userInfoServiceClient =
    new ClusterClient(LOGGER, cfg -> {
        cfg.setClusterProvider(
            new ClusterConfigClusterProvider(
                "topology/user.info",
                clusterConfigClient, LOGGER));
        cfg.setTransport(
            new ApacheAsyncHttpTransport());
    });

public void start() {
    Router router = Router.router(vertex);
    router.get("/say/hello/:id").handler(ctx -> {
        String userId = ctx.request().getParam("id");

        URI urlBuilder =
            new RequestUrlBuilder("/user")
                .appendToPath(userName).build();

        Request request = new Request("GET", urlBuilder);
        userInfoServiceClient
            .sendAsync(request).thenAccept(
                result -> {
                    byte[] body = result
                        .response().content().buffer();

                    JsonObject response = new JsonObject()
                        .put("name", userName)
                        .put("friends", new JsonString(body));

                    ctx.response().end(response.toBuffer());
                });
    }
}
```



Шаг 2.

Добавляем дочерний запрос к sendEmail и смотрим удалённую трассировку.

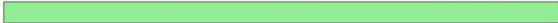

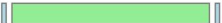
```
public void start() {
    Router router = Router.router.vertx();
    router.get("/user/:id").handler(ctx -> {
        String userId = ctx.request().getParam("id");

        try (TraceContextScope contextScope =
            beginNewTraceContextScope(context)) {
            try (ISpanWriter spanWriter =
                tracing.begin(TraceSpanKind.SERVER)) {
                spanWriter
                    .recordAnnotation(
                        Annotation.REQUEST_URL, "/user/<id>");
                spanWriter
                    .recordAnnotation(
                        Annotation.REQUEST_HOST, "localhost:8091");

                ctx.response().end(
                    new JsonObject()
                        .put("friends",
                            new JsonArray()
                                .add("vasya")
                                .add("petya")).toBuffer());

                spanWriter
                    .recordAnnotation(
                        Annotation.RESPONSE_CODE, "200");
            }
        }
    });
}
```

Распределенная трассировка

<div>Clear</div>										
Chains 0		Group 0		Traces 0 - 1 of 1						
service	url	host	request	response	code	cum	duration ↓	trace timeline		
Say-Hello-Service						1.5 sec	1.5 sec	33ff858e7f31451392e83b15059ab10c	2018-03-15T16:05:32.175	view logs
										
User.Info.Service	<div>GET</div>	K1707035	0	227	200	1.5 sec	0.9 sec			
Send.Email.Service	<div>POST</div>	K1707035	241	62	200		0.6 sec			

Итого

- ✓ Масштабирование сервисов снижает риск даунтайма для клиента
- ✓ Упрощение каждого отдельного сервиса снижает стоимость разработки системы в целом
- ✓ Мониторинг сервиса как единого целого снижает стоимость обслуживания системы
- ✓ Профит!