Туториал: мастер-экземляр платформы ("балансировщик" нагрузки)

Балансировка нагрузки Лямбда-Мю 3

Балансировка нагрузки Лямбда-Мю 3 – метод распределения запросов между несколькими экземплярами платформы Лямбда-Мю 3 с целью оптимизации использования ресурсов, резервирования ресурсов, сокращения времени обработки запросов, обеспечения отказоустойчивости и возможности масштабирования такого кластера.

Балансировка нагрузки может решать различные задачи и содержать специфические методы для конкретного проекта, например:

- ассиметричная нагрузка
- коммутация данных из одного формата в другой
- упаковку нескольких запросов от разных клиентов в один пакет запросов
- буферизация данных от медленных клиентов
- фильтрация запросов, не соответствующих протоколу
- проверка работоспособности экземпляров платформы
- аутентификация клиента
- пр.

Таким образом, «балансировщик» нагрузки может взять на себя дополнительные не дорогие в отношении ресурсов операции, которые позволят сосредоточить ресурсы экземпляров платформы на более приоритетных операциях.

Вопрос балансировки нагрузки нужно решать на ранних стадиях проекта. Балансировка непосредственно влияет на корректное, быстрое, отказоустойчивое взаимодействие между клиентом и платформой. Не принятые вовремя меры могут привести к серьёзным последствиям, в том числе материальным.

Пример реализации «балансировщика»

Платформа Лямбда-Мю 3 может использоваться как back end сервер веб-проекта. Одним из таких проектов является «Сомнология B2C» – мобильное приложение на Android. Продемонстрируем реализацию «балансировщика» этого проекта. Основными требованиями являются:

- проверка работоспособности экземпляров платформы
- возврат клиенту данных о наиболее предпочтительном сервере.

Эти требования будут реализованы дополнительным экземпляром платформы Лямбда-Мю 3. Реализация продемонстрирует взаимосвязи нескольких экземпляров платформы Лямбда-Мю 3 с меняющемся количеством логических машин и сторонних клиентов.

Репозитарий

...

Исходные данные

Проект «Сомнология B2C»:

- back end: платформа Лямбда-Мю 3 [server]
- front end: Android-приложение [client]

Нагрузка на [server] от разных [client] одинаковая.

Условия работоспособности [server]:

- к нему можно подключиться (корректно работает объект платформы TCP-server)
- при инициализации пользователя возвращается имя объекта Task (корректно запустилась логическая машина)
- корректный ответ на запрос о количестве клиентов

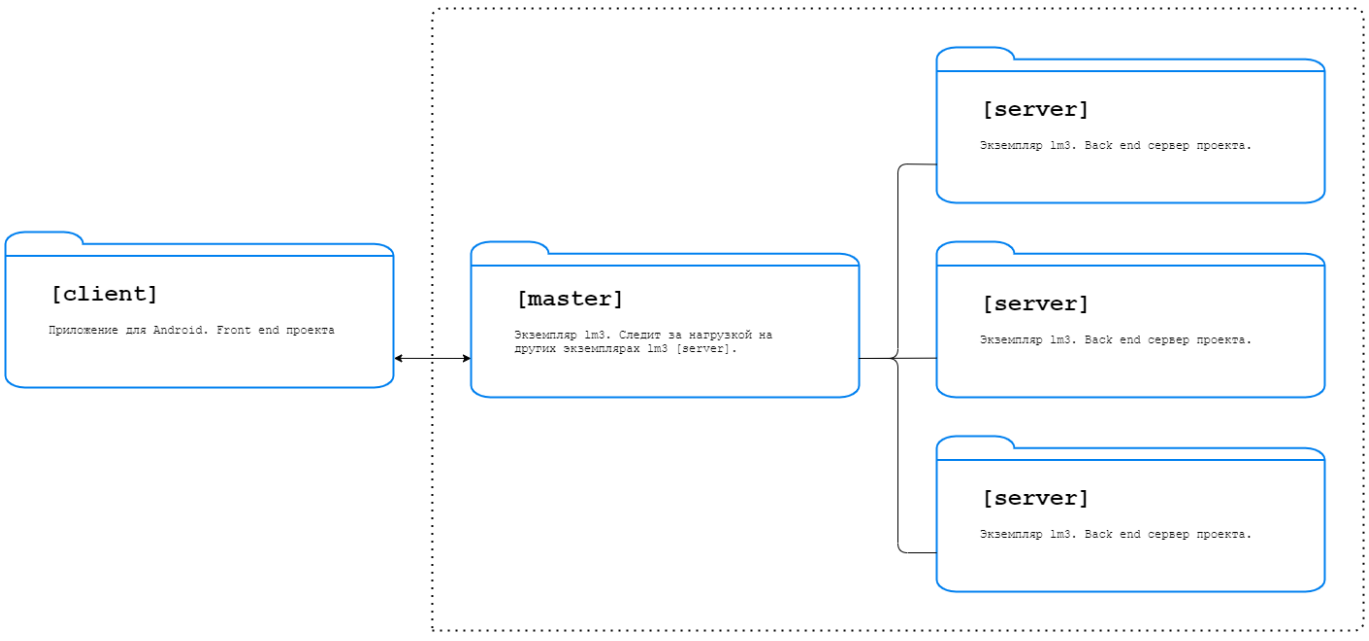
Для отладки решения будут использоваться следующие версии:

- [client] : Android-приложение с вырезанным пользовательским интерфейсом и бизнес-логикой
- [server] : платформа Лямбда-Мю 3 с вырезанной бизнес-логикой

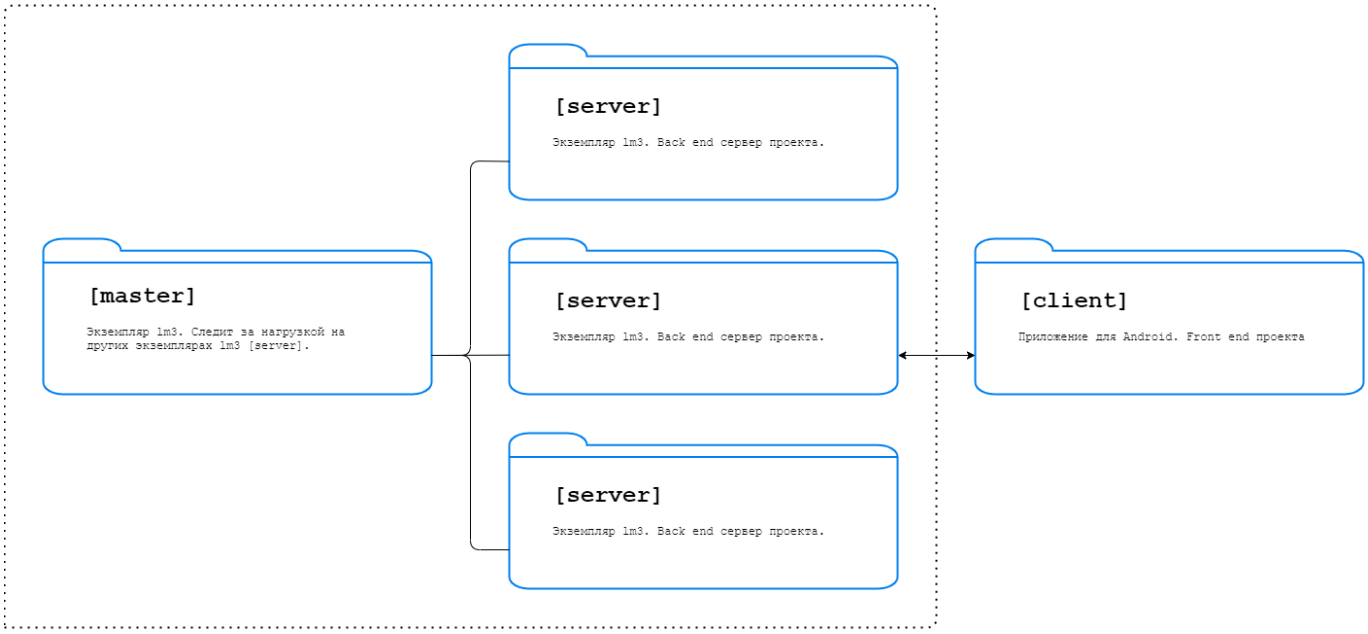
Цель проекта

Распределить нагрузку на экземпляры платформы [server]. При этом клиенты [client] будут сначала выполнять подключение к «балансировщику» [master], получать данные о наиболее подходящем экземпляре [server] и подключаться к нему.

Первый этап:



Второй этап:



Требования

Требованием к проекту является минимальные изменения исходного проекта [client] и [server]. Требование достигается использованием того же протокола, который разработан в исходном проекте.

[client] ↔ [master]

Запрос:

```
{ { Object = "Master", Command = "Event", Name = "DoSendServer" } }
```

Ответ:

```
{ { Object = "Master", Command = "Value", Name = "Server", Data =
<table[Result:string,[Data]:table[IP:string,Port:int]]> } }
```

[master] ↔ [server]

Запрос:

```
{ { Object = "Auth", Command = "Set", Name = "UserName", Data = <string>}, {
Object = "Auth", Command = "Set", Name = "UserMail", Data = <string> } }
```

Ответ:

```
{ { Object = "Auth", Command = "Value", Name = "TaskObject", Data = <string>
} }
```

Запрос:

```
{ { Object = "System", Command = "Get", Name = "UserCount" } }
```

Ответ:

```
{ { Object = "System", Command = "Value", Name = "UserCount", Data = <int> }
}
```

Файловая структура проекта



Взаимосвязи

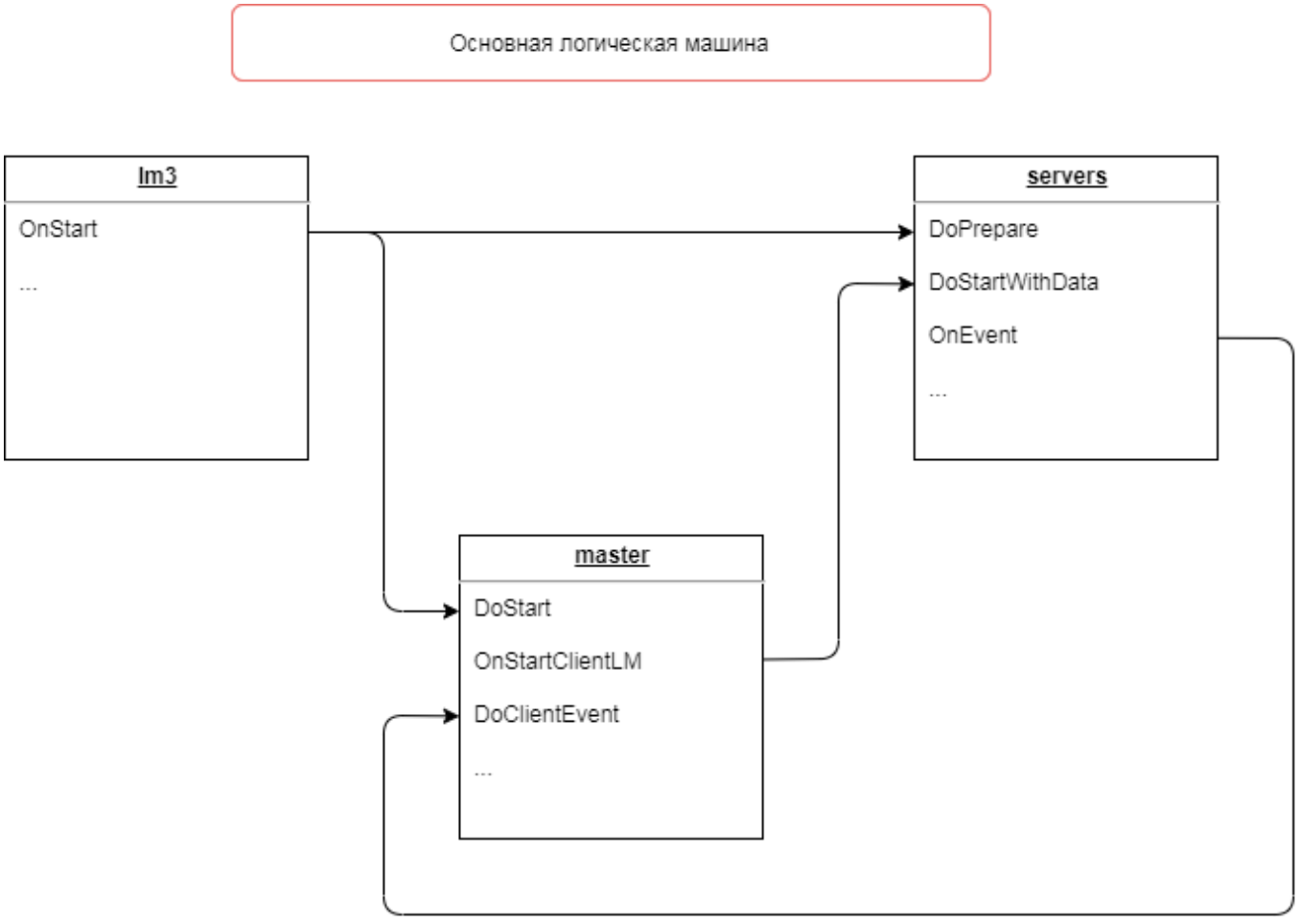
Объекты основной логической машины:

- TCP-server server: прием и передача данных клиенту
- DataPack packer: распаковка и запаковка данных клиенту
- Master master: обработка запросов клиентов и актуализация данных о экземплярах
- Servers servers (наследник LMList): взаимосвязь между ЛМ и объектом master
- Timer timer: по тику логируется существование хотя бы одного работающего экземпляра платформы из пула.

При старте основной логической машины

- вызывается DoPrepare объекта servers - проверяется на валидность конфигурация логических машин
- вызывается DoStart объекта master - цепочка ивентов, создающая для каждого экземпляра платформы (из конфигурационного файла основной логической машины) логическую машину с соответствующими параметрами
- активируется таймер timer

Диаграмма связей:



Взаимодействие с клиентом происходит следующим образом

- server получает запрос
- packer распаковывает данные
- master ищет наиболее подходящий экземляр
- packer пакует данные
- server отправляет ответ

Диаграмма связей:

From:
https://wiki.lambda-mu.com/ - **Lambda-Mu Wiki**

Permanent link:
https://wiki.lambda-mu.com/lm3/ce/t_master?rev=1606314658

Last update: **2020/11/30 14:12**