

Perl как AI-оркестратор: МСР-интеграция для сервисов агрегации

Комаров Никита

ВТБ, Специалист. САВРУС, Fullstack

Perl-Conf.Ru/26

Структура выступления

Perl-Conf.Ru/26

1. Обо мне
2. Введение
3. Проблема
4. Perl как сервис
5. Perl как AI-оркестратор
6. Вектор развития
7. Выводы





Комаров Никита

ВТБ, Специалист. САВРУС, Fullstack

- ▶ Программирую на Perl для души
- ▶ С Perl познакомился недавно, но есть что рассказать

- ▶ В первой части произведу 4 модернизации сервиса агрегации на понятной задаче в классической модели с Perl
- ▶ Во второй части добавим AI и посмотрим что из этого выйдет
- ▶ Сделаем выводы по каждой из частей



Классический вариант построения сервиса

Perl-Conf.Ru/26

Структура каталогов:

- ▶ lib/
- ▶ templates/
- ▶ public/
- ▶ t/
- ▶ script/
- ▶ my_app.conf

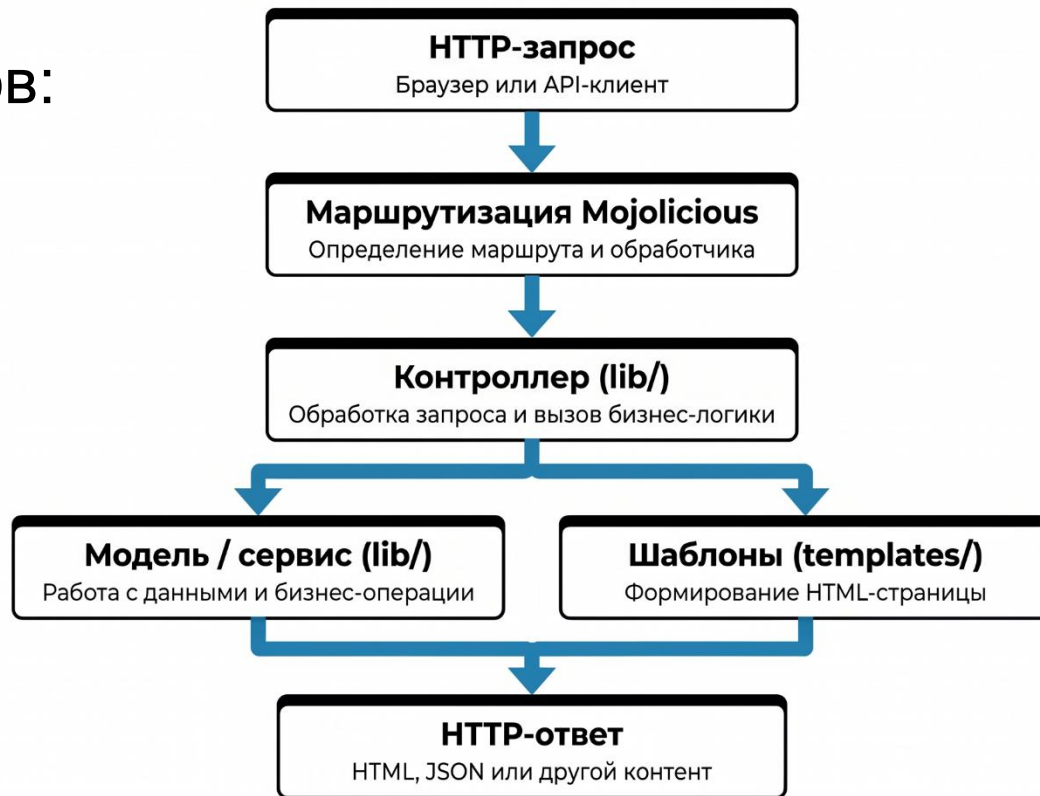


Рисунок 1 - Как взаимодействуют компоненты

Как было: агрегация в сервисе

Perl - Conf .Ru/26



Рисунок 2 - Архитектура до изменений

Задача: топ-10 магазинов за 365 дней

Transactions	
id	bigint
merchant_id	int
customer_id	int
amount	decimal(15, 2)
currency	char(3)
status	varchar(20)
created_at	datetime

Найти: топ-10 магазинов по
сумме успешных оплат за
последние 365 дней

В БД 10 млн записей

Рисунок 3 - Таблица транзакций

Perl считает журнал сам

```
1 SELECT merchant_id, amount
2 FROM transactions
3 WHERE created_at >= NOW() - INTERVAL 365 DAY
4 AND status = 'completed';
n
n+1 // ...
n+2
n+3 while (my $row = $sth->fetchrow_arrayref) {
n+4     my ($merchant_id, $amount) = @$row;
n+5     $result{$merchant_id}{count}++;
n+6     $result{$merchant_id}{amount} += $amount;
n+7 }
```

6 млн записей
попадет в хэш



Схема «до»: Perl - калькулятор

Perl - Conf .Ru/26



Рисунок 4 - Архитектура до изменений

Три роли Perl

Модуль занимается:

Perl-Conf.Ru/26



Perl

- ▶ Интеграцией
- ▶ Бизнес логикой
- ▶ Тяжелой аналитикой

Выполнение запроса **53.649 с**

Необходимо модернизировать существующий Perl-сервис

А если GROUP BY в MariaDB?

Perl - Conf . Ru / 26



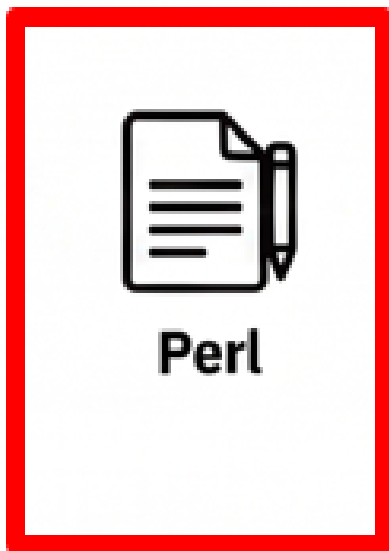
Рисунок 5 - Архитектура после изменений

Тот же ответ, всё ещё около минуты

```
1 SELECT merchant_id, COUNT(*), SUM(amount)
2 FROM transactions
3 WHERE status = 'completed' AND created_at >= NOW() - INTERVAL
4     365 DAY
5 GROUP BY merchant_id;
```

38.334 с < 53.649 с

Считать в приложении или в MariaDB - почти одно и то же, около минуты



Модуль занимается:

- ▶ Интеграцией
- ▶ Бизнес логикой
- ▶ Тяжелой аналитикой

Модернизируем существующий Perl-сервис



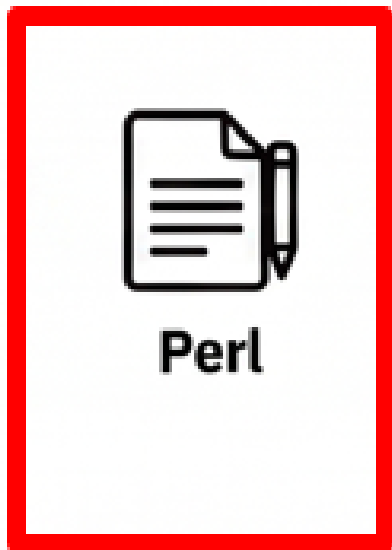
Рисунок 6 - Архитектура с ClickHouse без оптимизации кода

Модуль занимается:

- ▶ Интеграцией
- ▶ Бизнес логикой
- ▶ Тяжелой аналитикой

Дополнительно:

- ▶ Рядом с MariaDB стоит ClickHouse
- ▶ Выполнение запроса **116 с**



Поставить ClickHouse рядом без снятия с Perl тяжелой аналитики не ускоряет процесс

Добавление ClickHouse

Perl - Conf . Ru / 26

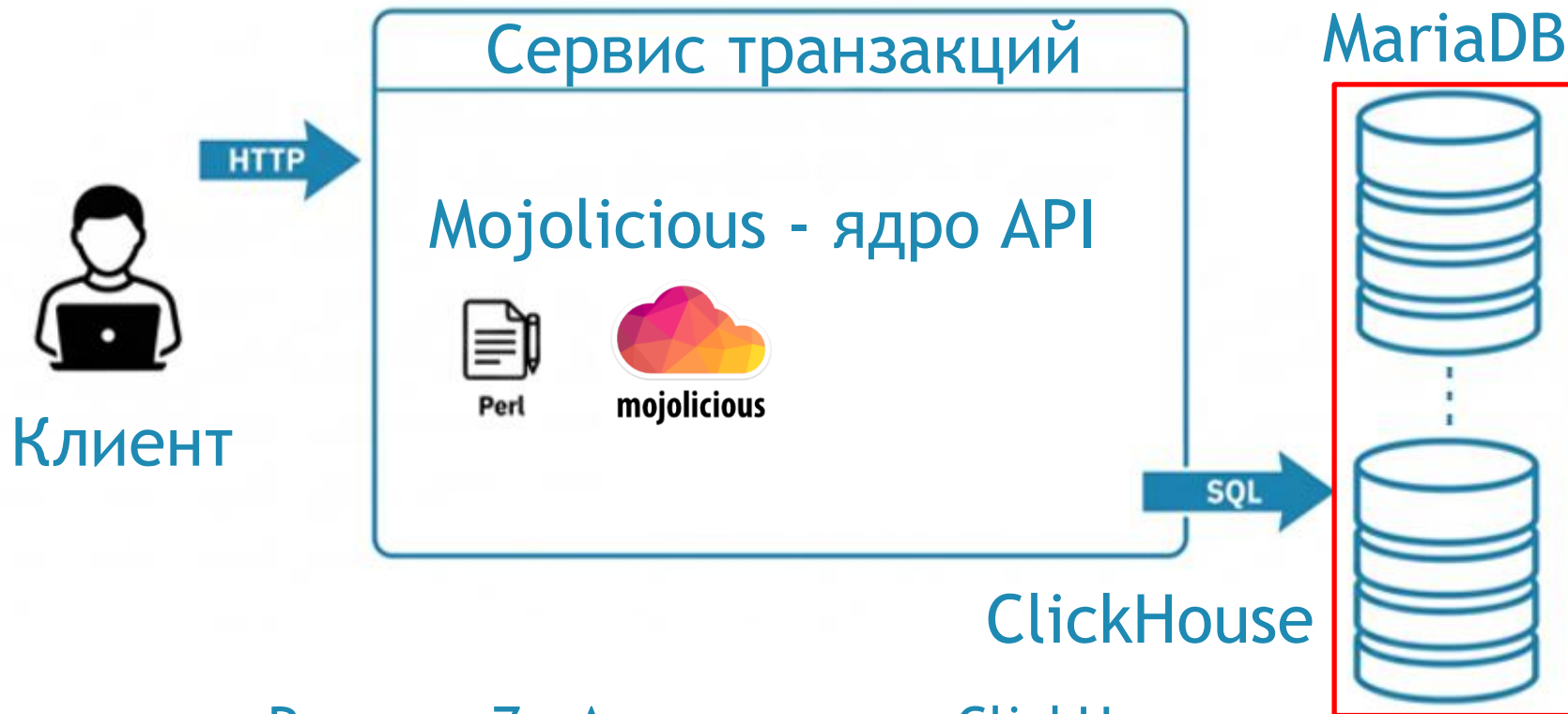
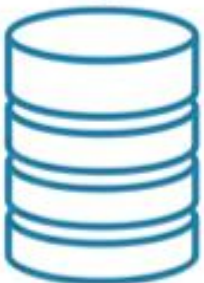


Рисунок 7 - Архитектура с ClickHouse

MariaDB



ClickHouse

- ▶ MariaDB занимается записью данных
- ▶ ClickHouse занимается аналитикой
- ▶ Раз в час ClickHouse инкрементально забирает новые данные из MariaDB
- ▶ Выполнение запроса на получение топ магазинов - **1,2 с**

- ▶ Читать в Perl - **54 с**, шесть миллионов строк в процессе. Это калькулятор
- ▶ GROUP BY в MariaDB - **38 с**. Место то же, касса, почти минута
- ▶ ClickHouse рядом, цикл тот же - **116 с**. Новый ящик без снятия счёта не ускоряет
- ▶ GROUP BY в ClickHouse - **1,2 с**, в Perl приходят **10** строк

- ▶ Понять запрос человека
- ▶ Выбрать автоматически нужный tool
- ▶ Perl отлично парсит текст
- ▶ Не писать SQL руками
- ▶ О, у Вас AI. Давай AI



Что такое MCP

Perl - Conf . Ru / 26

- ▶ Model Context Protocol
- ▶ Клиент вызывает сервер
- ▶ JSON-RPC 2.0
- ▶ Не база и не модель
- ▶ Сервер отдаёт каталог tools, клиент жмёт имя tool

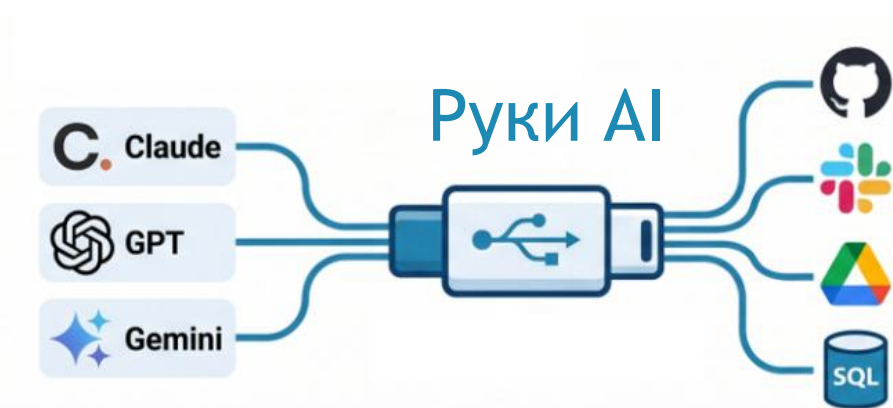


Рисунок 8 - Концепция MCP



- ▶ Готовый SDK от автора Mojolicious
- ▶ Собственная реализация

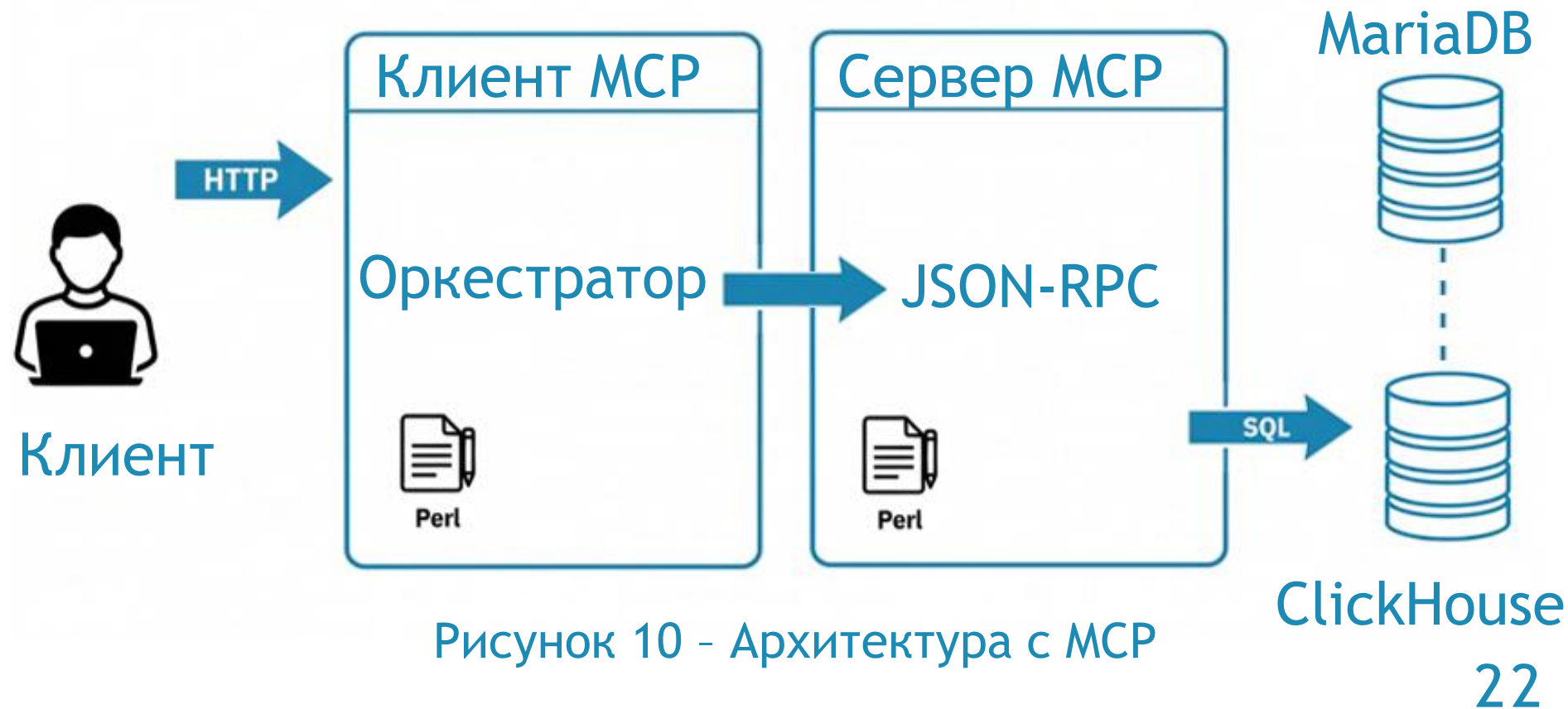
Воспользуемся вторым вариантом

Рисунок 9 - Готовый SDK



Внедрение MCP

Perl - Conf . Ru / 26



Пользователь пишет текст - инструмент выбирает

Perl - Conf .Ru/26

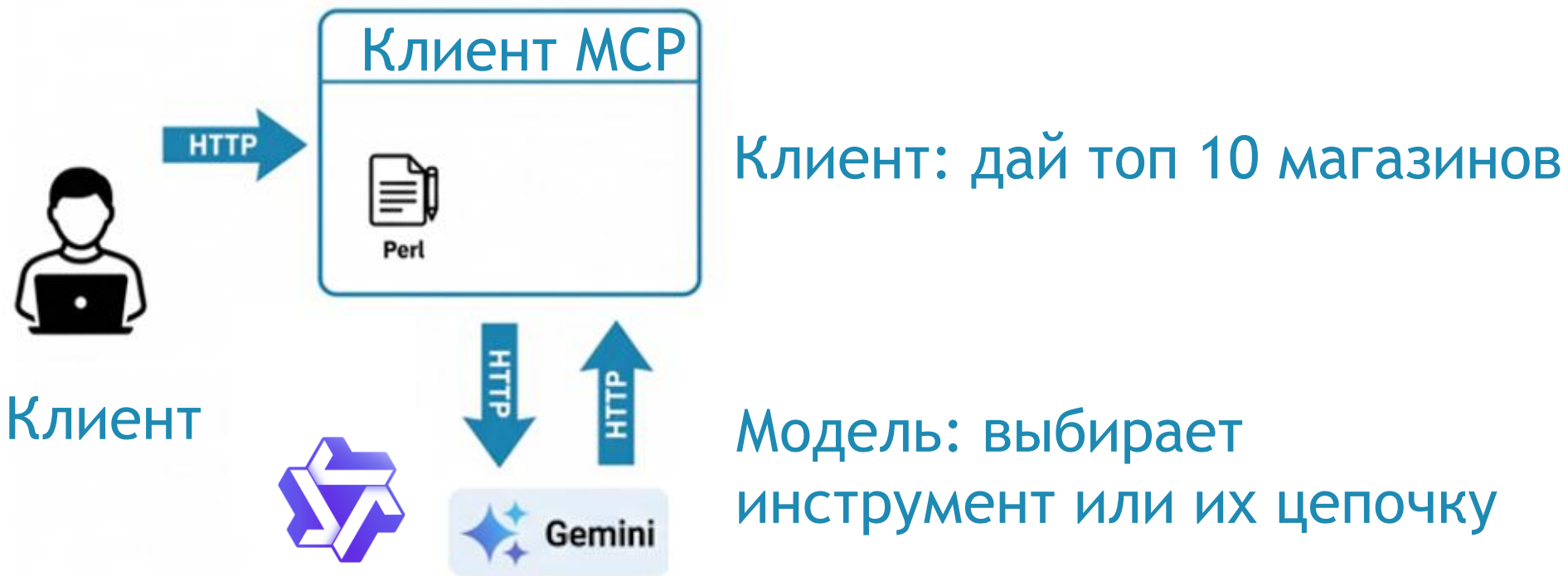
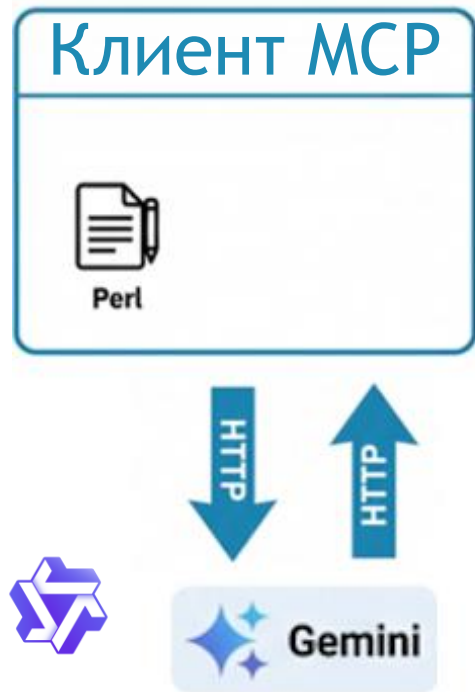


Рисунок 11 - Клиент и модель



- ▶ Mojo::UserAgent - POST к LLM
- ▶ Задаем роль, правила, доступные tools, ограничения, стиль в промт, разрешаем сказать не знаю и др.
- ▶ Получаем нужный tool и аргументы

Что ходит между клиентом и сервером

Perl - Conf .Ru/26

```
1{
2  "jsonrpc" : "2.0" ,
3  "id": "1",
4  "method": "tools/call",
5  "params":
6    {
7    "name": "get_merchant_totals",
8    "arguments": { "days": 365 }
9    }
10 }
```

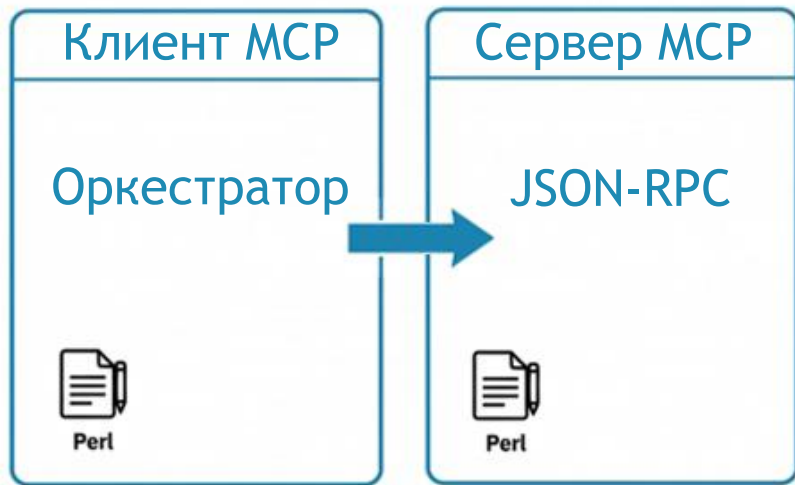


Рисунок 13 - Клиент MCP и сервер MCP

```
1 sub catalog {  
2   return [  
3     # get_merchant_totals  
4     # sync_new_transactions  
5     # get_pipeline_status  
n      ...  
n+1  ];  
n+2 }
```

- ▶ Содержат name, description, аргументы
- ▶ Клиент забирает через tools/list

```
m+1 if ($method eq 'initialize') { ... }  
m+2 if ($method eq 'tools/list') { ... catalog ... }  
m+3 if ($method eq 'tools/call') {  
m+4   my $data = call($params->{name}, ...);  
n     ...  
n+1 }
```



Что ходит между клиентом и сервером

```
1 {  
2  "jsonrpc": "2.0",  
3  "id": "1",  
4  "result": {  
5    "days": 365,  
6    "rows_in_perl": 10,  
7    "top": [  
8      {  
9        "merchant_id": 4707,  
10       "amount": 35246484  
11     }  
12   ]  
13 }  
14 }
```

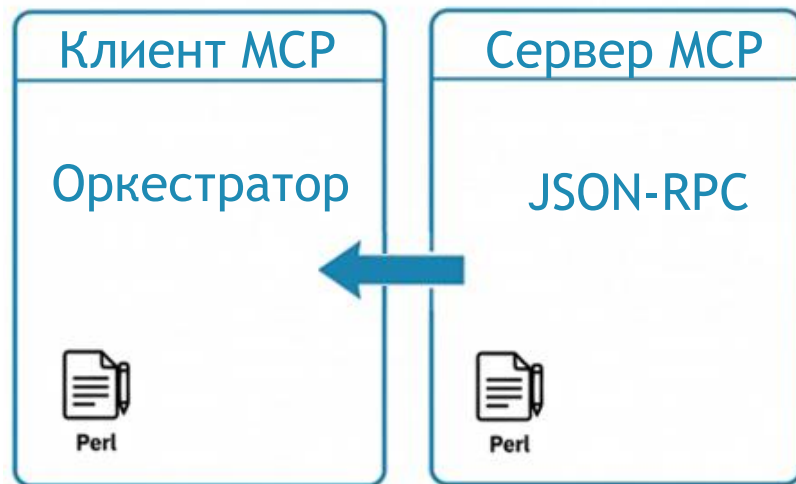
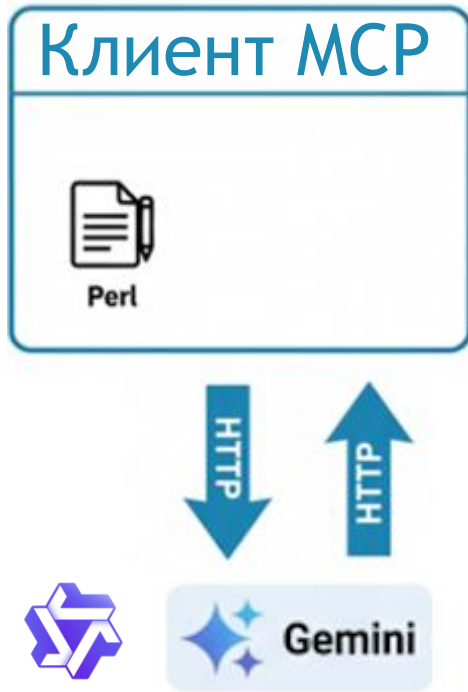


Рисунок 14 - Клиент MCP и сервер MCP



- ▶ Mojo::UserAgent - POST к LLM
- ▶ Задаем роль, правила, доступные tools, ограничения, стиль в промт, разрешаем сказать не знаю и др.
- ▶ Получаем обработанный результат и отдаем клиенту

- ▶ В первом эксперименте локально поднятая модель qwen3:0.6b выполняла запрос 130 секунд, из которых 86 секунд выбирала tools, 43 секунд писала ответ после получения результата от tool
- ▶ Во втором эксперименте через API gemini-3.8-flash выполнила код за 4 секунды
- ▶ В обоих случаях модели выбрали нужный инструмент - `get_merchant_totals`

Что происходит сейчас?

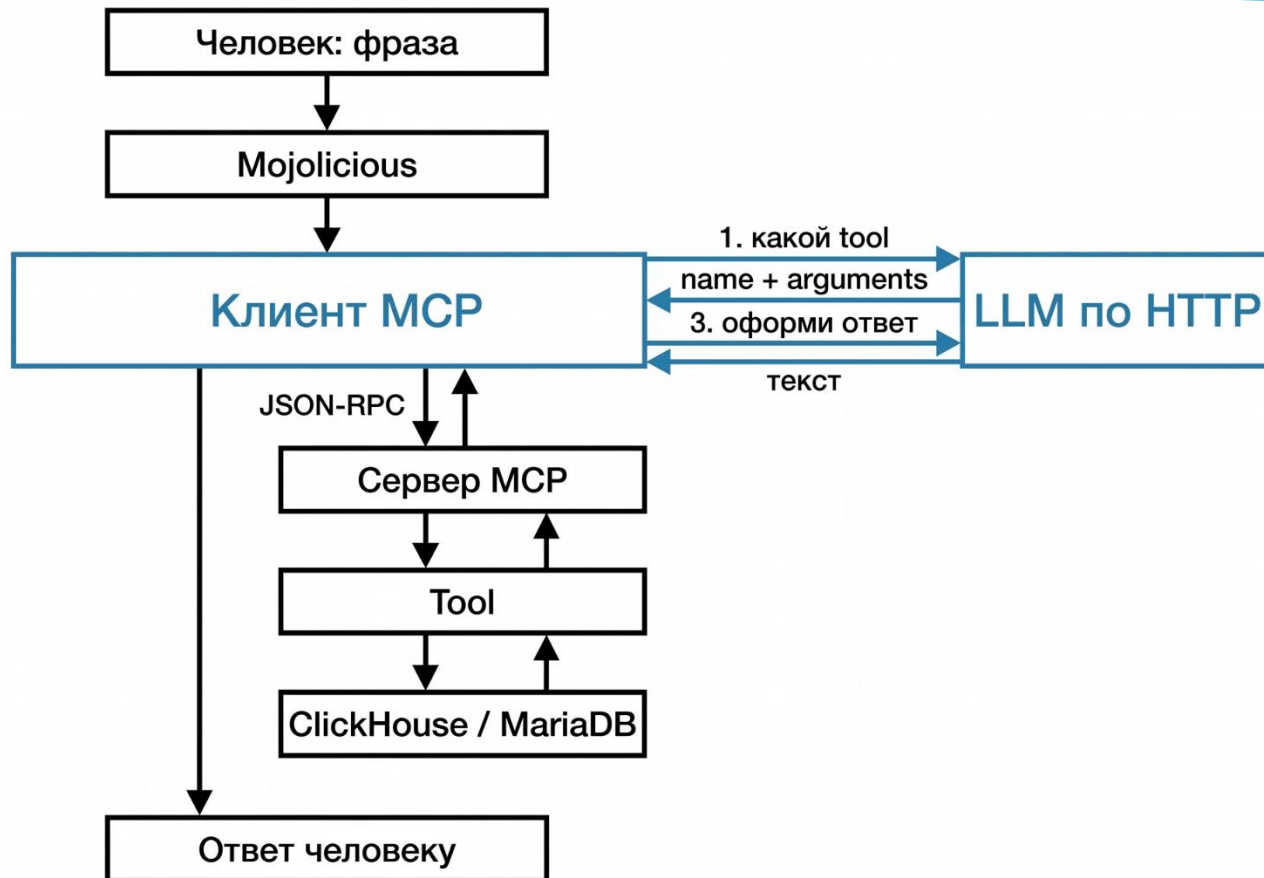


Рисунок 16 - Фиксированный сценарий

Можно ли это назвать оркестратором?

- ▶ Отсутствует цикл – один проход
- ▶ Отсутствует состояние между шагами
- ▶ Perl не решает, что делать дальше
- ▶ Отсутствуют условия остановки и обработки ошибок

Perl выполняет то, что решила модель

Добавим к рукам мозгов

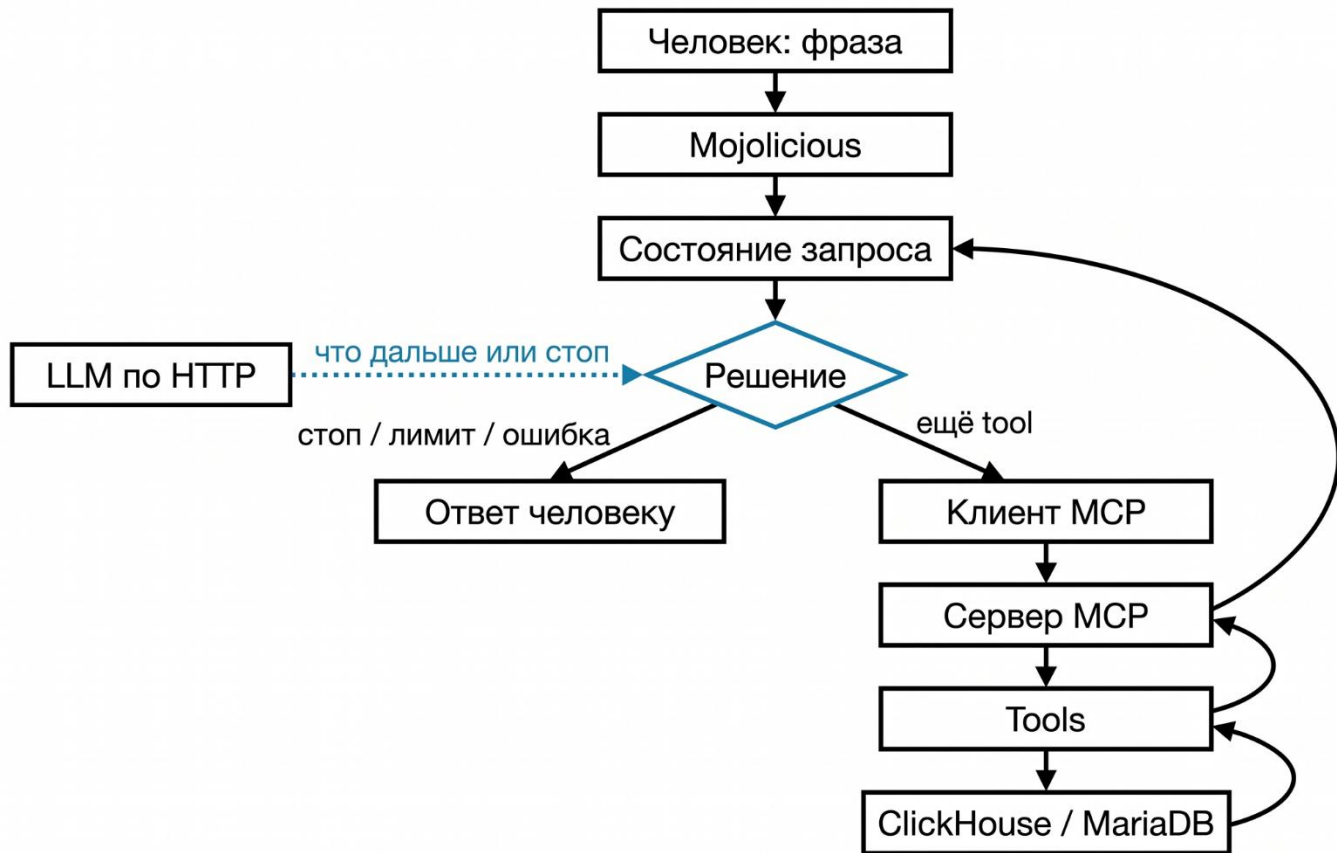


Рисунок 17 - Цикл вокруг состояния

Где Harness, а где оркестратор?

- ▶ AI-оркестратор стоит на harness: цикл, состояние, стоп
- ▶ В основе AI-оркестратора лежит Harness
- ▶ Кто вызывает - оркестратор
- ▶ Как крутится - Harness

Perl здесь оркестратор

Можно дать AI базу. Можно сделать меню без AI

- ▶ Человек пишет как коллеге
- ▶ Кнопки те же, безопасные
- ▶ Цепочка из вызовов без ручной склейки



Perl здесь оркестратор, а не калькулятор с доступом к БД

Как это может выглядеть у клиента

Perl-Conf.Ru/26

ФРАЗА	ИНСТРУМЕНТ
Топ мерчантов за 365 дней	
▼	
выполнить	

Рисунок 18 - Без AI

ФРАЗА	ИНСТРУМЕНТ
дай топ 10 сервисов	
выполнить	

Рисунок 19 - AI

Как это может выглядеть у клиента

Топ-10 сервисов за 365 дней

1. Сервис 4707 — 32 922 121,75 · 622 оплат
2. Сервис 2715 — 32 478 681,05 · 611 оплат
3. Сервис 58 — 32 405 696,72 · 619 оплат
4. Сервис 5670 — 32 205 227,01 · 643 оплат
5. Сервис 4356 — 32 199 072,51 · 600 оплат
6. Сервис 8925 — 32 188 303,66 · 630 оплат
7. Сервис 7583 — 32 096 378,77 · 623 оплат
8. Сервис 8936 — 32 093 299,10 · 607 оплат
9. Сервис 3587 — 32 061 902,96 · 630 оплат
10. Сервис 3923 — 32 043 620,61 · 610 оплат

Рисунок 20- Ответ от qwen3:0.6b

Мы рассмотрели модернизацию с AI и без AI:

- ▶ Если нужна высокая производительность, то придётся не только прикрутить ClickHouse, но и следить, чтобы код был оптимизирован. Рядом стоящий ClickHouse сам по себе отчёт не ускоряет
- ▶ Решение с MCP подходит, когда отчётов много: человек пишет фразу, модель выбирает tool, считает по-прежнему ClickHouse. Прямой доступ модель к БД не имеет

- ▶ Рассмотрен Perl как сервис
- ▶ Рассмотрен Perl как AI-оркестратор
- ▶ Представлена MCP интеграция
- ▶ Оценена производительность всех случаев на конкретной задаче





Сергей Жмылёв

Организатор Perl-Conf.Ru

Спасибо за внимание!

Рад ответить на Ваши вопросы

Perl-Conf.Ru/26