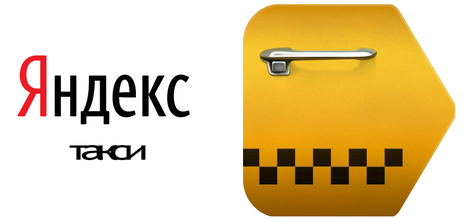
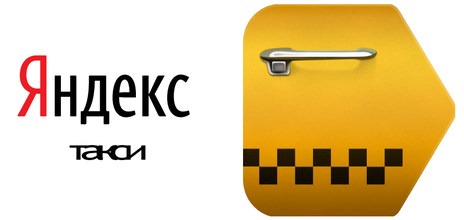


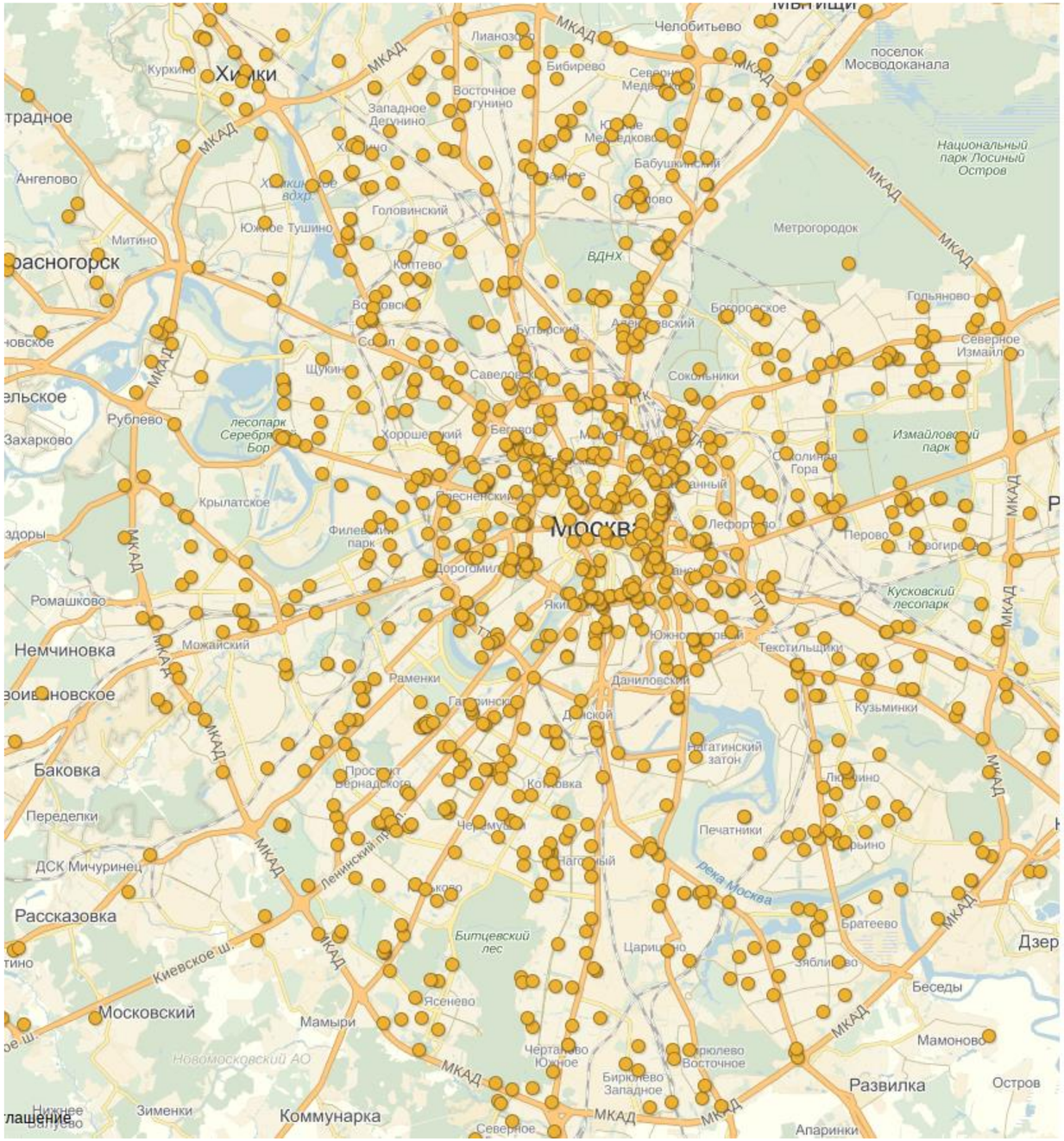
Асинхронность и многопоточность в Яндекс.Такси

Дмитрий Курилов

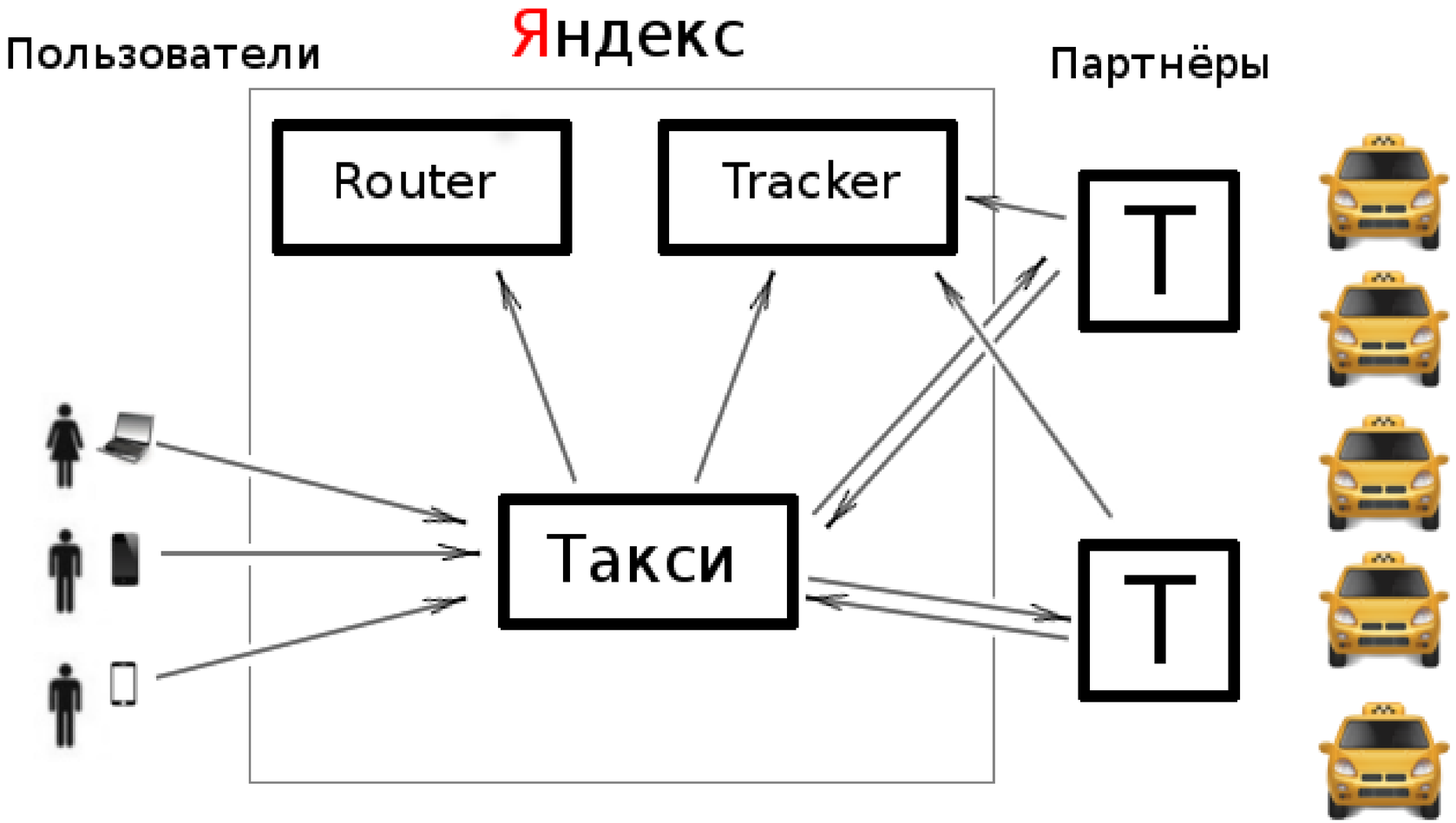


Яндекс.Такси

Когда нельзя ждать


МКАД
Химки
Куркино
Челобитьево
поселок Мосводоканала
Лианозово
Бибирево
Восточное Дегунино
Западное Дегунино
Национальный парк Лосиный Остров
Ангелово
Головинский
Южное Тушино
Митино
Метрогородок
ВДНХ
Коптево
Богородское
Гольяново
Сокольники
Северное Измайлово
Щукино
Рублево
Захарково
лесопарк Серебряный Бор
Беговой
Измайловский парк
Крылатское
Пресненский
Москва
Перово
Филевский парк
Дорогомилово
Ромашково
Кусковский лесопарк
Текстильщики
Немчиновка
Можайский
Раменки
Даниловский
Кузьминки
Баковка
Переделкино
Печатники
Переделки
ДСК Мичуринец
Рассказовка
Битцевский лес
Киевское ш.
Братеево
Беседы
Ясенево
Московский
Мамыри
Мамоново
Новомосковский АО
Развилка
Остров
Зименки
Коммунарка
Апаринки

АРХИТЕКТУРА



О ПРОЕКТЕ

- 6 500 — 10 000 водителей на линии
- > 200 партнёров
- > 200 000 заказов в неделю

СПЕЦИФИКА ПРОЕКТА

Интерактивно взаимодействуем с партнёрами и с внутренними сервисами Яндекса

Это обязывает:

- Быстро считать
- Жить с таймаутами
- И вообще жить без части партнёров и сервисов

ПРИМЕРЫ

ЯНДЕКС.ТРЕКЕР

- Предоставляет нам треки всех водителей
  (мы забираем сразу все координаты — 3k в gzip)
- Строит в памяти R-tree
- Регулярно и в отдельном треде (потому что 3k)

ЯНДЕКС.ТРЕКЕР

```
_drivers = {'dict': None, 'rtree': None}

@defer.inlineCallbacks
def update_drivers_positions():
    try:
        gzipped = yield fetch_all_tracks()
        (dct, rtree) = yield defer.deferToThread(
          extract_gzipped_tracks, gzipped)
        _drivers['dict'] = dct
        _drivers['rtree'] = rtree
    finally:
        callLater(UPDATE_INTERVAL, update_drivers_positions)
```

?

- Python?
- Многопоточность в Python? А как же GIL?

GIL

- Cpu-bound задачи — биндим, "отпуская" GIL
- Забиндив, выполняем вычисления в тредпуле
- Python CPU-bound код в треде выполнять нельзя

RELEASING GIL

```
import zlib
import base64

from twisted.internet import threads

def main():
    strings = ['%010000000d' % i for i in range(1000)]
    for string in strings:
        d = threads.deferToThread(compress, string)
        #d.addCallbacks(do_something_with_result)

def compress(string):
    return base64.b64encode(zlib.compress(string, 9))
```

RELEASING GIL

```
top - 17:50:13 up 35 days,  6:40,  2 users,  load average: 12.89, 8.27, 6.92
Tasks: 106 total,   1 running, 105 sleeping,   0 stopped,   0 zombie
Cpu0  : 95.7%us,  4.0%sy,  0.0%ni,  0.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  : 91.1%us,  6.2%sy,  0.0%ni,  1.6%id,  0.0%wa,  0.0%hi,  1.0%si,  0.0%st
Cpu2  : 97.3%us,  2.3%sy,  0.0%ni,  0.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  : 95.4%us,  3.6%sy,  0.0%ni,  0.7%id,  0.0%wa,  0.0%hi,  0.3%si,  0.0%st
Cpu4  : 98.3%us,  1.7%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu5  : 97.0%us,  2.7%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.3%si,  0.0%st
Cpu6  : 94.3%us,  4.3%sy,  0.0%ni,  0.7%id,  0.0%wa,  0.0%hi,  0.7%si,  0.0%st
Cpu7  : 94.1%us,  2.3%sy,  0.0%ni,  3.0%id,  0.0%wa,  0.0%hi,  0.7%si,  0.0%st
Cpu8  : 91.1%us,  3.6%sy,  0.0%ni,  1.0%id,  0.0%wa,  0.0%hi,  4.3%si,  0.0%st
Cpu9  : 81.9%us,  5.4%sy,  0.0%ni,  5.0%id,  0.0%wa,  0.0%hi,  7.7%si,  0.0%st
Cpu10 : 91.3%us,  2.0%sy,  0.0%ni,  2.0%id,  0.0%wa,  0.0%hi,  4.7%si,  0.0%st
Cpu11 : 87.4%us,  4.7%sy,  0.0%ni,  3.0%id,  0.0%wa,  0.0%hi,  5.0%si,  0.0%st
Cpu12 : 89.4%us,  4.3%sy,  0.0%ni,  1.0%id,  0.0%wa,  0.0%hi,  5.3%si,  0.0%st
Cpu13 : 92.3%us,  4.0%sy,  0.0%ni,  1.0%id,  0.0%wa,  0.0%hi,  2.7%si,  0.0%st
Cpu14 : 88.0%us,  3.7%sy,  0.0%ni,  2.3%id,  0.0%wa,  0.0%hi,  6.0%si,  0.0%st
Cpu15 : 93.3%us,  3.3%sy,  0.0%ni,  0.3%id,  0.0%wa,  0.0%hi,  3.0%si,  0.0%st
Mem:  99000288k total, 64896912k used, 34103376k free,  3468616k buffers
Swap:        0k total,        0k used,        0k free, 13095908k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
16873 dmkurilo  20   0 2522m 2.1g 3188 S 1261  2.2   2:49.06 python
15828 www-data  20   0  463m 197m 9484 S   11  0.2  68:29.58 twistd
```

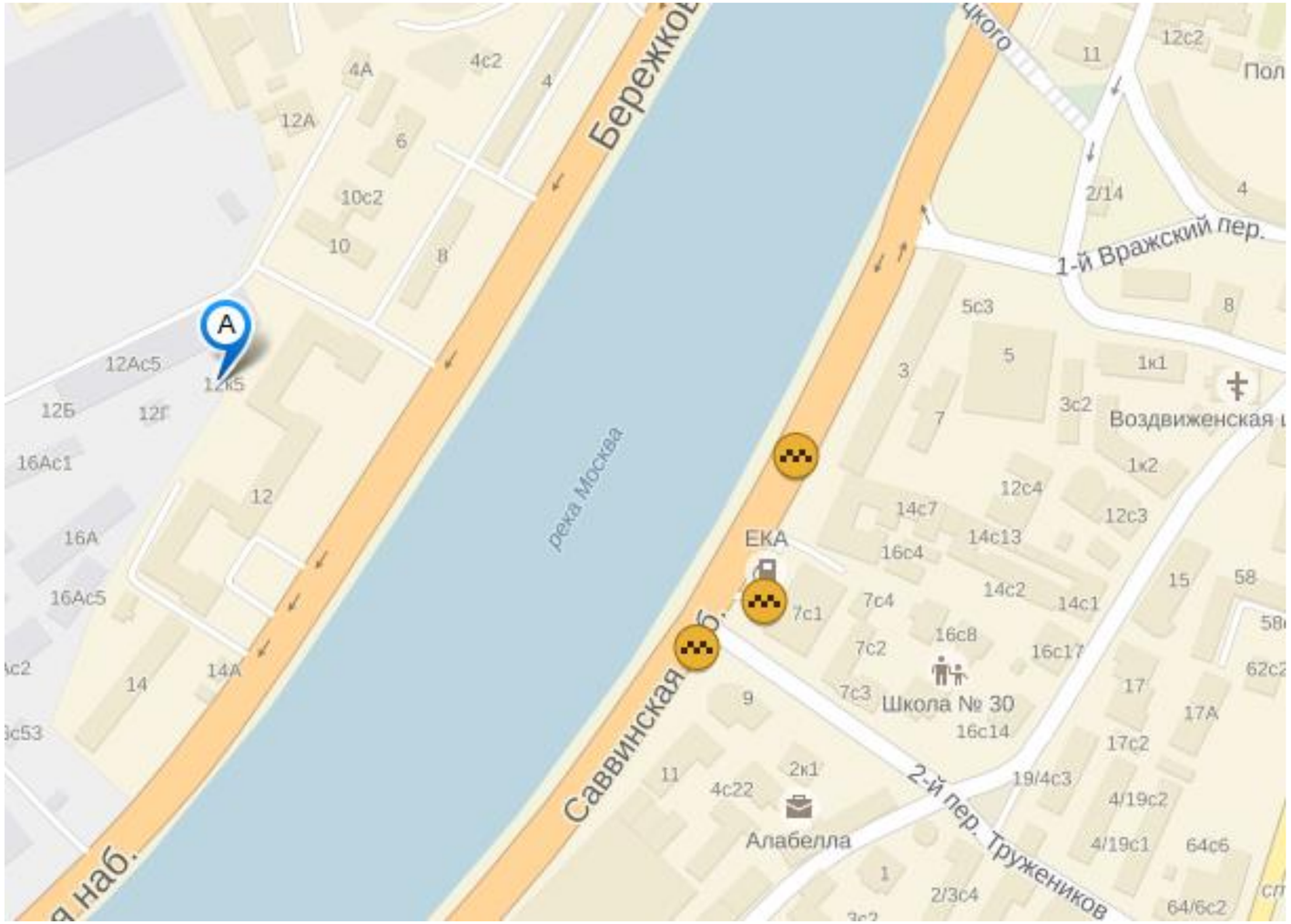
В ЦИФРАХ

- В одном потоке 129,82 s
- В нескольких потоках 20,51s
- Speedup 6,3 (на 16 ядрах, 8 из которых — HTT)

В СУХОМ ОСТАТКЕ

- GIL вносит некоторые неудобства
- Но обойти можно

ЯНДЕКС.РОУТЕР



ЯНДЕКС.РОУТЕР

- Строит маршрут между двумя точками
- Запрашиваем 4-90 маршрутов на заказ
- Ограничиваем число запросов к роутеру через очередь в памяти
- Считаем расстояние напрямую, когда роутер недоступен

ЯНДЕКС.РОУТЕР

```
from router import build_route

@defer.inlineCallbacks
def find_suitable_drivers(src_point):
    nearest = find_nearest_drivers(src_point, NEAREST_LIMIT)
    estimates = yield defer.DeferredList(
        [build_route(driver_point, src_point)
        for (driver_point, _) in nearest])
    suitable = []
    for ((driver_point, driver), estimate) in zip(nearest, estimates):
        (succeed, data) = estimate
        if succeed:
            (distance, time) = data
        else:
            (distance, time) = predict_route(driver_point, src_point)
        suitable.append((time, distance, driver))
    return sorted(suitable)
```

ДРУГИЕ ВНУТРЕННИЕ СЕРВИСЫ ЯНДЕКСА

- Обеспечивают поиск адресов, определение местоположения пользователя, помогают рассчитывать стоимость поездки заранее
- Мы умеем жить без большинства из внутренних сервисов, деградируя по функциональности
- Цель - научиться переживать падение любого из них (кроме трекера)

ПАРТНЕРЫ

- Какая-то их часть регулярно лежит
- Более 10% запросов отваливается по таймауту
- Водители заняты не только нашими заказами
- Поэтому запрашиваем, пока не найдём

ПАРТНЕРЫ

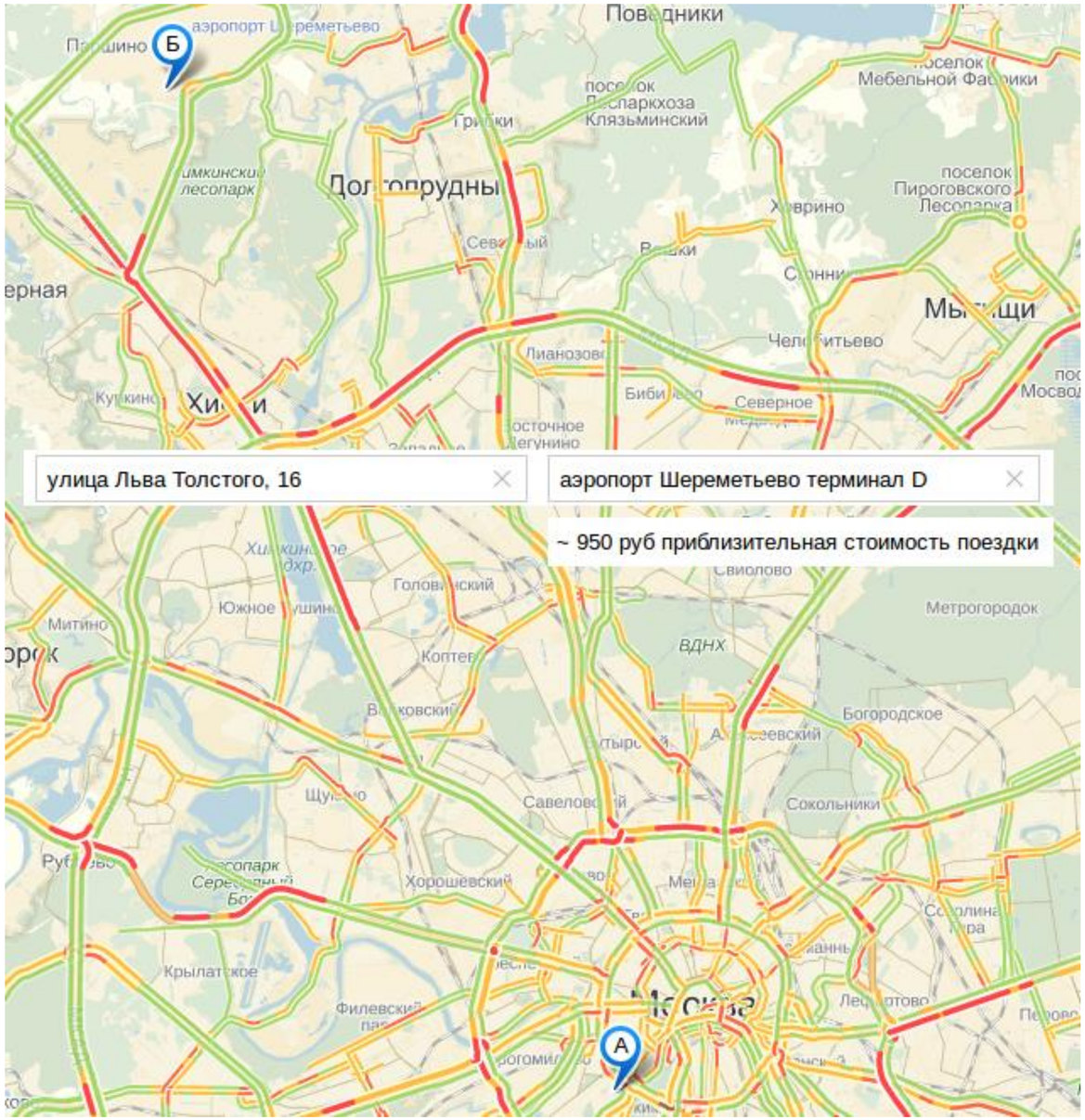
```
@defer.inlineCallbacks
def request_suitable_drivers(order_id, src_point):
    while (yield not is_order_expired(order_id)):
        suitable = yield find_suitable_drivers(src_point)
        for (park, request) in (yield build_requests(suitable)):
            reactor.callLater(0, send_request, park, request)

        del suitable
        del request

        yield sleep(REQUEST_INTERVAL)
```

НЕ ДОШЛИ РУКИ

КАЛЬКУЛЯТОР ТАРИФОВ



КАЛЬКУЛЯТОР ТАРИФОВ

- Для решения задачи необходимо определять, каким полигонам принадлежит точка
- Число точек ~ 10000
- Сейчас ищем за линейное время
- Хотим искать за логарифмическое

РАБОТА С MEMCACHE

- Для синхронизации данных между рантаймами Python используем memcache
- Запись/чтение занимают 200-50000 мкс
- Сейчас работаем с memcache в блокирующем режиме
- Хотим написать асинхронный драйвер

РАБОТА С MONGODB

- Нет драйвера для Twisted
- Поэтому работаем с MongoDB через тредпул
- Свой драйвер не осилим
- Хотим завернуть в отдельный тредпул для блокирующего кода, работающего с i/o-bound событиями

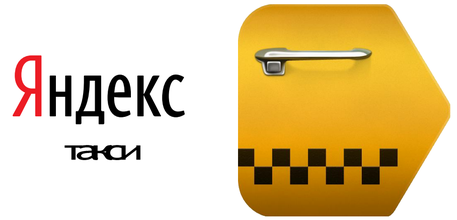
Я НЕ УСПЕЛ РАССКАЗАТЬ

- Об особенностях профайлинга асинхронных и многопоточных приложений
- Об их тестировании
- И о много другом...

В следующий раз обязательно расскажу :-)

РЕЗЮМЕ

- В интерактивном взаимодействии важно уметь жить с таймаутами и ошибками "на другом конце провода" и важно хорошо утилизировать CPU
- Мы пишем на Python/Twisted, но язык здесь роли не играет, специфика проявит себя в любом другом языке



Спасибо за внимание!
Вопросы?

Дмитрий Курилов

dmkurilov@yandex-team.ru